Introduction to PHP

A PHP file may contain text, HTML tags and scripts. Scripts in a PHP file are executed on the server.

What is PHP?

PHP: Hypertext Preprocessor, or PHP, is an open-source, server-side programming language. PHP is specifically designed to fill the gap between static HTML pages and fully dynamic pages, such as those generated through CGI code. PHP is embedded directly in the HTML source code.

What is a PHP File?

* PHP files may contain text, HTML tags and scripts
* PHP files are returned to the browser as plain HTML
* PHP files have a file extension of “.php”

Basic PHP Syntax

A PHP scripting block always starts with <?php and ends with ?>. A PHP scripting block can be placed anywhere in the document.

NOTE: on servers with shorthand support enabled you can start a scripting block with <? and end with ?>.

However, for maximum compatibility, you should use the standard form (<?php) rather than the shorthand form.

|  |
| --- |
| <?php  ?> |

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

Below, an example of a simple PHP script that sends the text “Hello World” to the browser:

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="utf-8">  <title>PHP Template</title>  </head>  <body>  <header>  <h1>PHP Template</h1>  </header>  <section>  <?php  echo "Hello World ";  ?>  </section>  <footer>  <p>&copy; Copyright 2015 UC Berkeley MySQL/PHP class</p>  </footer>  </body>  </html> |

Each code line in PHP must end with a semicolon. The semicolon is a separator and is used to distinguish one set of instructions from another.

There are two basic statements to output text with PHP: **echo** and **print**.

"echo" and "print" are not functions, they are PHP programming language construct. A programming language construct refers to a built-in feature of a programming language. The "echo" and "print" statements are nearly identical, but they have some differences. For example, the print statement returns a value of 1 if it is successful or a value of 0 if it is not successful, while the echo statement does not return a value.

Case Sensitivity in PHP

Unlike JavaScript, PHP is case insensitive. You can use any of the following versions of the echo statement without receiving an error message:

|  |
| --- |
| <?php  echo "<p>Line one. <br />";  Echo "Line two.<br />";  ECHO " and Line three. </p>";  ?> |

NOTE: Exceptions to PHP’s case insensitivity include variable and constant names, which are case sensitive.

Adding Comments to a PHP Script

Comments are lines you place in your code that do not get executed, but provide helpful information.

PHP supports two kinds of comments: line comments and block comments.

A line comment automatically terminates at the end of the line in which it is inserted. To create a line comment, add either two forward slashes (//) or the pound symbol (#) before the text you want to use as a comment. The // or # characters instruct the scripting engine to ignore all text immediately following the characters to the end of the line.

Block comments allow multiple lines of comment text to be added. You create a block comment by adding a forward slash and an asterisk (/\*) before the start of the text that you want included in the block, and adding an asterisk and a forward slash (\*/) after the last characters in the block. Any text or lines between the opening /\* characters and the closing \*/ characters are ignored by the PHP engine.

|  |
| --- |
| <html>  <body>  <?php  echo "<p>Print something </p>"; //This is a line comment  #This is also a line comment  /\*  This is  a comment block  \*/  ?>  </body>  </html> |

Using Variables and Constants

A variable is actually a specific location in the computer’s memory. To use a variable in a program, you first have to write a statement that creates the variable and assigns it a name.

Naming Variables

The name you assign to a variable is called an identifier.

* Identifiers must begin with a dollar sign ($).
* Identifiers may contain uppercase and lowercase letters, numbers, or underscores (\_). The first character after the dollar sign must be a letter.
* Identifiers cannot contain spaces.
* Identifiers are case sensitive.

Declaring and Initializing Variables

Before you can use a variable in your code, you have to create it. The process of specifying and creating a variable name is called declaring the variable. The process of assigning a first value to a variable is called initializing the variable. Some programming languages allow you to first declare a variable without initializing it. However, in PHP, you must declare and initialize a variable in the same statement.

All variables in PHP start with a $ sign symbol. Variables may contain strings, numbers, or arrays.

Below, the PHP script assigns the string "Hello World " to a variable called $txt:

|  |
| --- |
| <html>  <body>  <?php  $txt = "Hello World ";  echo $txt;  ?>  </body>  </html> |

To concatenate two or more variables together, use the dot (.) operator:

|  |
| --- |
| <html>  <body>  <?php  $txt1 = "Hello World ";  $txt2 = "Joe ";  echo $txt1 . " " . $txt2;  ?>  </body.  </html> |

The output of the script above will be: "Hello World Joe ".

Defining Constants

A constant contains information that does not change during the course of program execution. You can think of a constant as a variable with a static value.

Constant names do not begin with a dollar sign ($). In addition, it is common practice to use all uppercase letters for constant names. To declare and initialize a constant, you must use the define() function.

define("CONSTANT\_NAME", value);

define("DEFAULT\_LANGUAGE", "English");

define("VOTING\_AGE", 18);

By default, constant names are case sensitive, as are variables.

NOTE: you can make constant names case insensitive by passing a Boolean value of TRUE as a third argument to the define() function.

define("DEFAULT\_LANGUAGE", "English", TRUE);

NOTE: Standard programming convention is to use all uppercase letters for constant names, so you should avoid making your constant names case insensitive.

To display constant, you can pass a constant name to the echo statement in the same manner as variables.

echo "<p>The legal voting age is ", VOTING\_AGE, ".</p>";

echo "<p>The legal voting age is VOTING\_AGE.</p>";

|  |
| --- |
| <html>  <body>  <?php  define("VOTING\_AGE", 18);  echo "<p>The legal voting age is ", VOTING\_AGE, ".</p>";  ?>  </body>  </html> |

Working with Data Types

A data type is the specific category of information that a variable contains.

PHP supports five primitive data types:

|  |  |
| --- | --- |
| Data Type | Description |
| Integer numbers | The set of all positive and negative numbers and zero, with no decimal places. |
| Floating –point numbers | Positive or negative numbers with decimal places or numbers written using exponential notation. |
| Boolean | A logical value of “true” or “false” |
| String | Text such as "Hello World" |
| NULL | An empty value, also referred to as a NULL value |
|  |  |

The PHP language also supports reference, or composite, data types, which can contain multiple values or complex types of information.

The two reference data types supported by the PHP language are arrays and objects.

PHP is a loosely typed programming language. Loose typing is also known as dynamic typing because the data type for a variable can change after it has been declared.

|  |
| --- |
| $ChangingVariable = "Hello World"; // String  $ChangingVariable = 8; // Integer number  $ChangingVariable = 5.367; // Floating-point number  $ChangingVariable = TRUE; // Boolean  $ChangingVariable = NULL; // NULL |

NOTE: by contrast, C, C++, Java are strongly typed programming languages. Strong typing is also known as static typing because the data type for a variable will not change after it has been declared.

Numeric Data Types

PHP supports two numeric data types: integers and floating-point numbers.

Integers are positive and negative numbers and zero, with no decimal places (-250, -13, 0, 2, 6, 10, 100 and 10000).

A floating-point number contains decimal places or is written in exponential notation.

Exponential notation, or scientific notation, is a shortened format for writing very large numbers or numbers with many decimal places (-6.16, -4.4, 3.17, .52, 10.5, 2.0e11, 3.0E10).

Boolean Values

A Boolean value is a value of “true” or “false”.

You can use the integer value 1 to indicate a Boolean value of TRUE and 0 to indicate a Boolean value of FALSE.

In PHP programming, you can only use the words TRUE or FALSE to indicate Boolean values. PHP then converts the values TRUE and FALSE to the integers 1 and 0.

Arrays

An array is a set of data represented by a single variable name.

You use arrays when you want to store groups or lists of related information in a single, easily managed location.

There are three different kind of arrays:

* Numeric array – an array with a numeric ID key
* Associative array – an array where each ID key is associated with a value
* Multidimensional array – an array containing one or more arrays

Declaring and Initializing Indexed Arrays

In PHP, you can create numerically indexed arrays and associative arrays.

An element refers to a single piece of data that is stored within an array.

An index is an element’s numeric position within the array.

NOTE: By default, the numbering of elements within a PHP array starts with an index number of zero (0).

You create an array using the array() construct or by using the array name and brackets.

$array\_name = array(data0, data1, data2, data3);

Or

$array\_name[] = data0;

$array\_name[] = data1;

$array\_name[] = data2;

$array\_name[] = data3;

NOTE: unlike in variables, the preceding statements in arrays do not overwrite the existing values. Instead, each value is assigned to the $array\_name[] array as a new element using the next consecutive index number.

|  |
| --- |
| $names = array("Peter", "Queenie", "Joe"); |

|  |
| --- |
| $names[] = "Peter";  $names[] = "Queenie";  $names[] = "Joe"; |

Note: Most programming languages require that all elements in an array be of the exact same data type. In PHP, the values assigned to different elements of the same array can be of different data types.

$HotelReservation = array(

"My Name", // guest name (string)

2, // # of nights (integer)

89.95, // price per night (floating-point)

true); // nonsmoking room (Boolean)

Accessing Element Information

To find out the total number of elements in an array, use the count() function. You pass to the count() function the name of the array whose elements you want to count.

|  |
| --- |
| echo "<p>Number of elements in HotelReservation array is: ", count($HotelReservation), "</p>"; |

PHP includes the print\_r(), var\_export(), and var\_dump() functions, which you can use to display or return information about variables. These functions are most useful with arrays because they display the index and value of each element. You pass to each function the name of an array (or other type of variable).

Print\_r($MyArray);

The print\_r() function does not include any HTML formatting tags, so the array elements are displayed as a continuous string of text. To display the array elements on individual lines, use <pre> tags.

|  |
| --- |
| echo "<p><pre>";  print\_r($MyArray);  echo "</pre></p>"; |

Modifying Elements

You modify values in existing array elements in the same fashion as you modify values in a standard variable, except that you include the index for an individual element of the array.

$MyArray[2] = "More data";

$MyArray[1] = "Less data";

Declaring and Initializing Associative Arrays

PHP creates indexed arrays by default with a starting index of 0. With associative arrays, you can use any alphanumeric keys that you want for the array elements. You specify an element’s key by using the array operator (=>) in the array() construct.

$array\_name = array(key => value, key2 => value2, …);

You can also use the following syntax to assign key values to an associative array by using array names and brackets. NOTE: when using this syntax, you use the standard assignment operator (=) and not the array operator (=>).

$array\_name["key3"] = "value3";

$array\_name["key4"] = "value4";

To refer to an element in an associative array, you place an element’s key in single or double quotation marks inside the array brackets.

|  |
| --- |
| echo "<p> The value of key3 is {$array\_name['key3']}.</p>\n";  echo "<p> The value of key4 is " . $array\_name["key4"] . ".</p>\n"; |

In this example we use an array to assign birthday to the different persons:

|  |
| --- |
| $bd = array("Peter "=> "Mar 20 ", "Queenie "=> "Apr 10 ", "Joe "=> "Jul 3 "); |

This example is the same, but shows a different way of creating the array:

|  |
| --- |
| <?php  $bd["Peter "] = "Mar 20 ";  $bd["Queenie "] = "Apr 10 ";  $bd["Joe "] = "Jul 3 ";  echo "Queenie birthday is: " . $bd["Queenie "];  ?> |

The code above will output:

|  |
| --- |
| Queenie birthday is: Apr 10 |

Multidimensional Arrays

In the multidimensional array, each element in the main array can also be an array. And each element in the sub-array can be an array, and so on.

A multidimensional array with automatically assigned ID keys:

|  |
| --- |
| $families = array("Peter"=>array("Martin", "Lois", "Mary"),  "Queenie"=>array("James"),  "Joe"=>array("Amy", "Bill", "Jan")); |

The array should look like this:

|  |
| --- |
| array  (  ["Peter"] => array  (  [0] => "Martin"  [1] => "Lois"  [2] => "Mary"  )  ["Queenie"] => array  (  [0] => "James"  )  ["Joe"] => array  (  [0] => "Amy"  [1] => "Bill"  [2] => "Jan"  )  ) |

PHP Operators

Operators are used to operate on values.

Arithmetic Operators

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

Assignment Operators

|  |  |  |
| --- | --- | --- |
| Operator | Example | Is The Same As |
| = | $x = $y; | $s = $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; |
|  |  |  |

Comparison Operators

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| == | Is equal to | (5 == 8) return false |
| != | Is not equal | (5 != 8) return true |
| > | Is greater than | (5 > 8) return false |
| >= | Is greater than or equal to | (5 >= 8) return false |
| < | Is less than | (5 < 8) return true |
| <= | Is less than or equal to | (5 <= 8) true |
|  |  |  |

Logical Operators

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

Conditional Statements

* 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.
* elseif statement: is used when the if … else statement to execute a set of code if one of several condition are true.

Syntax

|  |
| --- |
| if (condition) {  code to be executed if condition is true;  } else {  code to be executed if condition is false;  } |

The following example will output “Have a nice weekend!” if the current day is Friday, otherwise it will output “Have a nice day!”:

|  |
| --- |
| <html>  <body>  <?php  $d = date("D");  if ($d == "Fri") {  echo "Have a nice weekend! ";  } else {  echo "Have a nice day! ";  }  ?>  </body>  </html> |

PHP Switch Statement

The switch statement in PHP is used to perform one of several different actions based on one of several different conditions.

NOTE: 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 none of the above;  } |

To rewrite the above elseif statement:

|  |
| --- |
| <html>  <body>  <?php  $d = date("D");  switch ($d) {  case "Fri":  echo "Have a nice weekend! ";  break;  case "Sun":  echo "Have a nice Sunday! ";  break;  default:  echo "Have a nice day! ";  }  ?>  </body>  </html> |

References:

a). Copied from <http://www.w3schools.com/php/default.asp>