PHP Looping

Looping statements in PHP are used to execute the same block of code a specified number of times.

In PHP we have the following looping statements:

* while: loops through a block of code if and 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.
* for: loops through a block of code a specified number of times.
* foreach: loops through a block of code for each element in a array.

The while Statement

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

Syntax

|  |
| --- |
| while (condition) {  code to be executed;  } |

The following example demonstrates a loop that will continue to run as long as the variable i is less than, or equal 5.

|  |
| --- |
| <html>  <body>  <?php  $i = 1;  while ($i <= 5) {  echo "The number is " . $i . "<br>";  $i++;  }  ?>  </body>  </html> |

The do…while 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); |

|  |
| --- |
| <html>  <body>  <?php  $i = 1;  do {  echo "The number is " . $i . "<br>";  $i++;  } while ($i <= 5);  ?>  </body>  </html> |

The for Statement

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

Syntax

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

|  |
| --- |
| <html>  <body>  <?php  for ($i =1; $i <= 5; $i++) {  echo "The number is " . $i . "<br>";  }  ?>  </body>  </html> |

The foreach Statement

The foreach statement is used to loop through arrays.

For every loop, the value of the current array element is assigned to $value (and the array pointer is moved by one) – so on the next loop, you will be looking at the next element.

Syntax

|  |
| --- |
| foreach ($array as $value) {  code to be executed;  } |

The following example demonstrates a loop that will print the value of the given array:

|  |
| --- |
| <html>  <body>  <?php  $arr = array("one", "two", "three", "four", "five");  foreach ($arr as $value) {  echo "Value: " . $value . "<br>";  }  ?>  </body>  </html> |

PHP Functions

A function is a block of code that can be executed whenever we need it.

NOTE: in PHP, there are more than 700 built-in functions available.

Creating PHP function:

* All functions start with the word "function()"
* Name the function – it should be possible to understand what the function does by its name. (The name can start with a letter or underscore (not a number).
* Add a "{" – the function code starts with the opening curly brace
* Insert the function code
* Add a "}" – the function is finished by a closing curly brace.

|  |
| --- |
| <html>  <body>  <?php  function HelloWorld() {  echo "Hello World! ";  }  HelloWorld();  ?>  </body>  </html> |

The output of the code above will be:

|  |
| --- |
| Hello World! |

PHP Functions – Adding parameters

We can add parameters to a function. A parameter is just like a variable.

(The parameters are specified inside the parentheses. If you have more than one parameter, use comma as a separator)

|  |
| --- |
| <html>  <body>  <?php  function HelloWorld\_with\_Name($MyName) {  echo "Hello World! " . " My name is " . $MyName;  }  HelloWorld\_with\_Name("John Smith");  ?>  </body>  </html> |

The output of the code above will be:

|  |
| --- |
| Hello World! My name is John Smith |

PHP Functions – Return values

Functions can also be used to return values.

|  |
| --- |
| <html>  <body>  <?php  function add($x, $y) {  $total = $x + $y;  return $total;  }  echo "1 + 16 = " . add(1, 16)  ?>  </body>  </html> |

The output of the code above will be:

|  |
| --- |
| 1 + 16 = 17 |

References:

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