LOOPS

Often when you write code, you want the same block of code to run over and over again a certain number of times. So, instead of adding several almost equal code-lines in a script, we can use loops.

Loops are used to execute the same block of code again and again, as long as a certain condition is true.

In PHP, we have the following loop types:

- 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 following chapters will explain and give examples of each loop type.

The PHP while Loop

The while loop executes a block of code as long as the specified condition is true.

Example

Print $i as long as $i is less than 6:

```
$i = 1;

while ($i < 6) {

  echo $i;

  $i++;
```

```
}
```

The while loop does not run a specific number of times, but checks after each iteration if the condition is still true.

The condition does not have to be a counter, it could be the status of an operation or any condition that evaluates to either true or false.

The break Statement in while loop

With the break statement we can stop the loop even if the condition is still true:

Example

Stop the loop when $i is 3:

```
$i = 1;

while ($i < 6) {

  if ($i == 3) break;

  echo $i;

  $i++;

}
```

The continue Statement in while loop

With the continue statement we can stop the current iteration, and continue with the next:

Example

Stop, and jump to the next iteration if $i is 3:

```
$i = 0;

while ($i < 6) {
```

```
  $i++;
  if ($i == 3) continue;
  echo $i;
}
```

Alternative Syntax in while loop

The while loop syntax can also be written with the endwhile statement like this

Example

Print $i as long as $i is less than 6:

```
$i = 1;
while ($i < 6):
  echo $i;
  $i++;
endwhile;
```

The PHP do...while Loop

The do...while loop will always execute the block of code at least once, it will then check the condition, and repeat the loop while the specified condition is true.

Example

Print $i as long as $i is less than 6:

```
$i = 1;


do {
```

```
  echo $i;
  $i++;
} while ($i < 6);
```

Note: In a do...while loop the condition is tested AFTER executing the statements within the loop. This means that the do...while loop will execute its statements at least once, even if the condition is false. See example below.

Let us see what happens if we set the $i variable to 8 instead of 1, before execute the same do...while loop again:

Example

Set $i = 8, then print $i as long as $i is less than 6:

```
$i = 8;

do {
  echo $i;
  $i++;
} while ($i < 6);
```

The code will be executed once, even if the condition is never true.

The break Statement in do..while loop

With the break statement we can stop the loop even if the condition is still true:

Example

Stop the loop when $i is 3:

```php
$i = 1;


do {
  if ($i == 3) break;
  echo $i;
  $i++;
} while ($i < 6);
```

The continue Statement in do..while loop

With the continue statement we can stop the current iteration, and continue with the next:

Example

Stop, and jump to the next iteration if $i is 3:

```php
$i = 0;


do {
  $i++;
  if ($i == 3) continue;
  echo $i;
} while ($i < 6);
```

The PHP for Loop

The for loop is used when you know how many times the script should run.

Syntax

```
for (expression1, expression2, expression3) {
  // code block
}
```

This is how it works:

- *expression1* is evaluated once
- *expression2* is evaluated before each iterarion
- *expression3* is evaluated after each iterarion

Example

Print the numbers from 0 to 10:

```
for ($x = 0; $x <= 10; $x++) {
  echo "The number is: $x <br>";
}
```

Example Explained

1. The first expression, $x = 0;, is evaluated once and sets a counter to 0.
2. The second expression, $x <= 10;, is evaluated *before* each iteration, and the code block is only executed if this expression evaluates to true. In this example the expression is true as long as $x is less than, or equal to, 10.
3. The third expression, $x++;, is evaluated *after* each iteration, and in this example, the expression increases the value of $x by one at each iteration.

The break Statement in for loop

With the break statement we can stop the loop even if the condition is still true:

Example

Stop the loop when $i is 3:

```
for ($x = 0; $x <= 10; $x++) {
  if ($i == 3) break;
  echo "The number is: $x <br>";
}
```

The continue Statement in for loop

With the continue statement we can stop the current iteration, and continue with the next:

Example

Stop, and jump to the next iteration if $i is 3:

```
for ($x = 0; $x <= 10; $x++) {
  if ($x == 3) continue;
  echo "The number is: $x <br>";
}
```

PHP foreach Loop

The foreach loop - Loops through a block of code for each element in an array or each property in an object.

The foreach Loop on Arrays

The most common use of the foreach loop, is to loop through the items of an array.

Example

Loop through the items of an indexed array:

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


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

For every loop iteration, the value of the current array element is assigned to the variabe $x. The iteration continues until it reaches the last array element.

Keys and Values

The array above is an indexed array, where the first item has the key 0, the second has the key 1, and so on.

Associative arrays are different, associative arrays use named keys that you assign to them, and when looping through associative arrays, you might want to keep the key as well as the value.

This can be done by specifying both the key and value in the foreach defintition, like this:

Example

Print both the key and the value from the $members array:

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

```php
foreach ($members as $x => $y) {
  echo "$x : $y <br>";
}
```

The foreach Loop on Objects

The foreach loop can also be used to loop through properties of an object:

Example

Print the property names and values of the $myCar object:

```php
class Car {
  public $color;
  public $model;
  public function __construct($color, $model) {
    $this->color = $color;
    $this->model = $model;
  }
}

$myCar = new Car("red", "Volvo");

foreach ($myCar as $x => $y) {
```

```
  echo "$x: $y <br>";
}
```

The break Statement in foreach loop

With the break statement we can stop the loop even if it has not reached the end:

Example

Stop the loop if $x is "blue":

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

foreach ($colors as $x) {
  if ($x == "blue") break;
  echo "$x <br>";
}
```

The continue Statement in foreach loop

With the continue statement we can stop the current iteration, and continue with the next:

Example

Stop, and jump to the next iteration if $x is "blue":

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

foreach ($colors as $x) {
  if ($x == "blue") continue;
```

```
  echo "$x <br>";
}
```

Foreach Byref

When looping through the array items, any changes done to the array item will, by default, NOT affect the original array:

Example

By default, changing an array item will not affect the original array:

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

foreach ($colors as $x) {
  if ($x == "blue") $x = "pink";
}

var_dump($colors);
```

BUT, by using the & character in the foreach declaration, the array item is assigned *by reference*, which results in any changes done to the array item will also be done to the original array:

Example

By assigning the array items *by reference*, changes will affect the original array:

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

foreach ($colors as &$x) {
```

```
  if ($x == "blue") $x = "pink";
}


var_dump($colors);
```

Alternative Syntax in foreach loop

The foreach loop syntax can also be written with the endforeach statement like this

Example

Loop through the items of an indexed array:

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


foreach ($colors as $x) :
  echo "$x <br>";
endforeach;
```

Break in For loop

The break statement can be used to jump out of a for loop.

Example

Jump out of the loop when $x is 4:

```
for ($x = 0; $x < 10; $x++) {
  if ($x == 4) {
    break;
```

```
  }
  echo "The number is: $x <br>";
}
```

Break in While Loop

The break statement can be used to jump out of a while loop.

Break Example

```
$x = 0;


while($x < 10) {
  if ($x == 4) {
    break;
  }
  echo "The number is: $x <br>";
  $x++;
}
```

Break in Do While Loop

The break statement can be used to jump out of a do...while loop.

Example

Stop the loop when $i is 3:

```
$i = 1;


do {
```

```
  if ($i == 3) break;
  echo $i;
  $i++;
} while ($i < 6);
```

Break in For Each Loop

The break statement can be used to jump out of a foreach loop.

Example

Stop the loop if $x is "blue":

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

foreach ($colors as $x) {
  if ($x == "blue") break;
  echo "$x <br>";
}
```

Continue in For Loops

The continue statement stops the current iteration in the for loop and continue with the next.

Example

Move to next iteration if $x = 4:

```
for ($x = 0; $x < 10; $x++) {
  if ($x == 4) {
    continue;
```

```
  }
  echo "The number is: $x <br>";
}
```

Continue in While Loop

The continue statement stops the current iteration in the while loop and continue with the next.

Continue Example

Move to next iteration if $x = 4:

```
$x = 0;

while($x < 10) {
  if ($x == 4) {
    continue;
  }
  echo "The number is: $x <br>";
  $x++;
}
```

Continue in Do While Loop

The continue statement stops the current iteration in the do...while loop and continue with the next.

Example

Stop, and jump to the next iteration if $i is 3:

```php
$i = 0;


do {
  $i++;
  if ($i == 3) continue;
  echo $i;
} while ($i < 6);
```

Continue in For Each Loop

The continue statement stops the current iteration in the foreach loop and continue with the next.

Example

Stop, and jump to the next iteration if $x is "blue":

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


foreach ($colors as $x) {
  if ($x == "blue") continue;
  echo "$x <br>";
}
```