PHP Loops

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal code-lines in a script, we can use loops to perform a task like this.
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

The PHP while Loop

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

```
while (condition is true) {
   code to be executed;
}
```

The example below first sets a variable $x to 1 ($x = 1). Then, the while loop will continue to run as long as $x is less than, or equal to 5 ($x <= 5). $x will increase by 1 each time the loop runs ($x++):
Example

```
<html>
<body>
<? php
$x = 1;
while($x <= 5) {
   echo "The number is: $x <br>";
   $x++;    }
?>
</body>
</html>
Output:
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

The PHP do...while Loop

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);
```

Syntax

The example below first sets a variable $x to 1 ($x = 1). Then, the do while loop will write some output, and then increment the variable $x with 1. Then the condition is checked (is $x less than, or equal to 5?), and the loop will continue to run as long as $x is less than, or equal to 5:

Example

```
<?php
$x = 1;
do {
   echo "The number is: $x <br>";
   $x++;
} while ($x <= 5);
?>
Output:
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

Notice that in a do while loop the condition is tested AFTER executing the statements within the loop. This means that the do while loop would execute its statements at least once, even if the condition is false the first time.

The example below sets the $x variable to 6, then it runs the loop, **and then the condition is checked**:

Example

```
<?php
$x = 6;
do {
   echo "The number is: $x <br>";
   $x++;
} while ($x<=5);
?>
Output: The number is: 6
```

The PHP for Loop

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

Syntax

for (*init counter; test counter; increment counter*) {
   code to be executed;
}

Parameters:

- *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 example below displays the numbers from 0 to 10:

Example

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

Output:

```
The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
The number is: 6
The number is: 7
The number is: 8
The number is: 9
The number is: 10
```

The PHP foreach Loop

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;
}
```

Syntax

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

The following example demonstrates a loop that will output the values of the given array ($colors):

Example

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

foreach ($colors as $value) {
    echo "$value <br>";
}
?>
Output:
red
green
blue
yellow
```

The real power of PHP comes from its functions; it has more than 1000 built-in functions.

PHP User Defined 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.

Create a User Defined Function in PHP

A user defined function declaration starts with the word "function":

```
function functionName() {
   code to be executed;
}
```

Syntax

Note: A function name can start with a letter or underscore (not a number).

Tip: Give the function a name that reflects what the function does!

Function names are NOT case-sensitive.

In the example below, we create a function named "writeMsg()". The opening curly brace ({) indicates the beginning of the function code and the closing curly brace (}) indicates the end of the function. The function outputs "Hello world!". To call the function, just write its name:

Example

```
<?php
function writeMsg() {
   echo "Hello world!";
}
writeMsg(); // call the function
?> output : Hello world!
```

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.
The following example has a function with one argument ($fname). When the familyName() function is called, we also pass along a name (e.g. Jani), and the name is used inside the function, which outputs several different first names, but an equal last name:

Example

```
<?php
function familyName($fname) {
   echo "$fname Refsnes.<br>";
}
familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>
```

Output:

```
Jani Refsnes.
Hege Refsnes.
Stale Refsnes.
Kai Jim Refsnes.
Borge Refsnes.
```

The following example has a function with two arguments ($fname and $year):

Example

```
<?php
function familyName($fname, $year) {
   echo "$fname Refsnes. Born in $year <br>";
}
familyName("Hege", "1975");
familyName("Stale", "1978");
familyName("Kai Jim", "1983");
?>
```

Output:

```
Hege Refsnes. Born in 1975
Stale Refsnes. Born in 1978
Kai Jim Refsnes. Born in 1983
```

PHP Default Argument Value

The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument:

Example

```
<?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);
?>
```

Output:

```
The height is : 350
The height is : 50
The height is : 135
The height is : 80
```

PHP Functions - Returning values

To let a function return a value, use the return statement:

Example

```
<?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);
?>
```

Output:

```
5 + 10 = 15
7 + 13 = 20
2 + 4 = 6
```