PHP if...else... elseif Statements

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

PHP Conditional Statements

Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this.

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 if...else... elseif Statements

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

Output "Have a good day!" if the current time (HOUR) is less than 20:

```
<?php
$t = date("H");

if ($t < "20") {
  echo "Have a good day!";
}
?>
```

PHP if...else... elseif Statements

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

Output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

```
<?php
$t = date("H");

if ($t < "20") {
  echo "Have a good day!";
} else {
```

PHP if...else... elseif Statements

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 first condition is false and this condition is true;
} else {
  code to be executed if all conditions are false;
}
```

Example

Output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

```
<?php
$t = date("H");

if ($t < "10") {
  echo "Have a good morning!";
```

PHP Switch Statement

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

The PHP switch Statement

Use the switch statement to **select one of many blocks of code to be executed**.

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;
```

PHP Switch Statement

This is how it works: First we have a single expression *n* (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use break to prevent the code from running into the next case automatically. The defaultstatement is used if no match is found.

Example

```
<?php
$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!";
```

PHP Loops

In the following chapters you will learn how to repeat code by using loops in PHP.

PHP 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.

PHP Loops

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 displays the numbers from 1 to 5:

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

PHP Loops

The PHP while Loop

This example counts to 100 by tens:

Example

```
<?php
$x = 0;
while($x <= 100) {
  echo "The number is: $x <br>";
  $x+=10;
}
?>
```

Example Explained

$x = 0; - Initialize the loop counter ($x), and set the start value to 0

$x <= 100 - Continue the loop as long as $x is less than or equal to 100

$x+=10; - Increase the loop counter value by 10 for each iteration

PHP Loops

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.

Syntax

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

PHP Loops

The PHP do...while Loop

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:

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

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.

PHP Loops

The PHP do...while Loop

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

```
<?php
$x = 6;

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

PHP Loops

The PHP for Loop

The for loop - Loops through a block of code a specified number of times.
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 for each iteration;
}
```

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

PHP Loops

The PHP for Loop

The example below displays the numbers from 0 to 10:

Example

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

Example Explained

$x = 0; - Initialize the loop counter ($x), and set the start value to 0

$x <= 10; - Continue the loop as long as $x is less than or equal to 10

$x++ - Increase the loop counter value by 1 for each iteration

PHP Loops

The PHP for Loop

This example counts to 100 by tens:

```
<?php
for ($x = 0; $x <= 100; $x+=10) {
  echo "The number is: $x <br>";
}
?>
```

Example Explained

$x = 0; - Initialize the loop counter ($x), and set the start value to 0

$x <= 100; - Continue the loop as long as $x is less than or equal to 100

$x+=10 - Increase the loop counter value by 10 for each iteration

PHP Loops

The PHP foreach Loop

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

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

Syntax

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

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.

PHP Loops

The PHP foreach Loop

The following example will output both the keys and the values of the given array ($age):

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $val) {
  echo "$x = $val<br>";
}
?>
```

You will learn more about arrays in the PHP Arrays chapter.