PHP Conditional Statements

While writing programs/scripts, there will be scenarios where you would want to execute a particular statement only if some condition is satisfied. In such situations we use Conditional statements.

In PHP, there are 4 different types of Conditional Statements.

* if statements
* if...else statements
* if...elseif...else statements
* switch statement

if statement:

When we want to execute some code when a condition is true, then we use if statement.

**Syntax:**

if(condition)

{

// code to be executed if 'condition' is true

}

**The if...else statement**

When we want to execute some code when a condition is true, and some other code when that condition is false, then we use the if...else pair.

**Syntax:**

if(condition)

{

// code to be executed if 'condition' is true

}

else

{

// code to be executed if 'condition' is false

}

**Example**,

<?php

$age = 26;

if($age <= 25)

{

echo "You are not allowed to consume alchohol";

}

else

{

echo "Enjoy the drinks";

}

?>

**Output:**

Enjoy the drinks

**The if...else...elseif statement**

When we want to execute different code for different set of conditions, and we have more than 2 possible conditions, then we use if...elseif...else pair.

**Syntax:**

if(condition1)

{

// code to be executed if 'condition1' is true

}

elseif(condition2)

{

// code to be executed if 'condition2' is true

}

else

{

/\* code to be executed if both 'condition1'

and 'condition2' are false \*/

}

**Example**,

<?php

// speed in kmph

$speed = 110;

if($speed < 60)

{

echo "Safe driving speed";

}

elseif($speed > 60 && $speed < 100)

{

echo "You are burning extra fuel";

}

else

{

// when speed is greater than 100

echo "Its dangerous";

}

?>

**output**

Its dangerous

In the example above, we have also used logical operator &&. Logical operators are very useful while writing multiple conditions together.

**The switch statement**

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

Using a switch statement, we can specify multiple conditions along with the code to be executed when that condition is true, thereby implementing a menu style program.

Syntax

switch(X) // X can be a variable or an expression.

{

case value1:

// execute this code when X=value1

break;

case value2:

// execute this code when X=value2

break;

case value3:

// execute this code when X=value3

break;

...

default:

/\* execute this when X matches none of

of the specified options \*/

}

Example

<?php

$car = "Jaguar";

switch($car)

{

case "Audi":

echo "Audi is amazing";

break;

case "Mercedes":

echo "Mercedes is mindblowing";

break;

case "Jaguar":

echo "Jaguar is the best";

break;

default:

echo "$car is Ok";

}

?>

**Output**

Jaguar is the best

**Iterative Statements/Loops**

A Loop is used to execute something over and over again.

PHP supports:

* while loop
* do… while loop
* for loop
* foreach loop

**PHP while Loop**

The while loop in PHP has two components, one is a condition and other is the code to be executed. It executes the given code until the specified condition is true.

Syntax:

<?php

while(condition)

{

/\*

execute this code till the

condition is true

\*/

}

?>

For example, let's take the problem mentioned in the beginning of this tutorial. Let's print numbers from 1 to 10.

<?php

$a = 1;

while($a <= 10)

{

echo "$a | ";

$a++; // incrementing value of a by 1

}

?>

**output**

1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

**PHP do...while Loop**

The do...while loop is a little different from all the loops in PHP because it will execute at least one time, even if the condition is false, can you guess how? Well because the condition is checked after the loop's execution, hence the first time when the condition is checked, the loop has already executed once.

Syntax:

<?php

do {

/\*

execute this code till the

condition is true

\*/

} while(condition)

?>

<?php

$a = 1;

do {

echo "$a | ";

$a++; // incrementing value of a by 1

} while($a <= 10)

?>

1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

<?php

$a = 11;

do {

echo $a;

$a++; // incrementing value of a by 1

} while($a <= 10)

?>

**PHP for Loop**

The for loop in PHP doesn't work like while or do...while loop, in case of for loop, we have to declare beforehand how many times we want the loop to run.

Syntax:

<?php

for(initialization; condition; increment/decrement)

{

/\*

execute this code till the

condition is true

\*/

}

?>

The parameters used have following meaning:

**initialization**: Here we initialize a variable with some value. This variable acts as the loop counter.

**condition**: Here we define the condition which is checked after each iteration/cycle of the loop. If the condition returns true, then only the loop is executed.

**increment/decrement**: Here we increment or decrement the loop counter as per the requirements.

**Example**

<?php

for($a = 1; $a <= 10; $a++)

{

echo "$a <br/>";

}

?>

**output**

1

2

3

4

5

6

7

8

9

10

**Nested for Loops**

We can also use a for loop inside another for loop. Here is a simple example of nested for loops.

<?php

for($a = 0; $a <= 2; $a++)

{

for($b = 0; $b <= 2; $b++)

{

echo "$b $a ";

}

}

?>

**output**

0 0

1 0

2 0

0 1

1 1

2 1

0 2

1 2

2 2

**PHP foreach Loop**

The foreach loop in PHP is used to access key-value pairs of an array. This loop only works with arrays and you do not have to initialise any loop counter or set any condition for exiting from the loop, everything is done implicitly(internally) by the loop.

Syntax:

<?php

foreach($array as $var)

{

/\*

execute this code for all the

array elements

$var will represent all the array

elements starting from first element,

one by one

\*/

}

?>

**Example:**

<?php

$array = array("Jaguar", "Audi", "Mercedes", "BMW");

foreach($array as $var)

{

echo "$var <br/>";

}

?>

**output**

Jaguar

Audi

Mercedes

BMW

**Arrays**

In PHP, there are three kinds of arrays:

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Numeric Arrays**

* A numeric array stores each array element with a numeric index.
* There are two methods to create a numeric array.

In the following example the indexes are automatically assigned (the index starts at 0):

$cars=array("Saab","Volvo","BMW","Toyota");

In the following example we assign the index manually:

$cars[0]="Saab";

$cars[1]="Volvo";

$cars[2]="BMW";

$cars[3]="Toyota";

Example

In the following example you access the variable values by referring to the array name and index:

<?php

$cars[0]="Saab";

$cars[1]="Volvo";

$cars[2]="BMW";

$cars[3]="Toyota";

echo $cars[0] . " and " . $cars[1] . " are Swedish cars.";

?>

The code above will output:

Saab and Volvo are Swedish cars.

**Associative Arrays**

* An associative array, each ID key is associated with a value.
* When storing data about specific named values, a numerical array is not always the best way to do it.
* With associative arrays we can use the values as keys and assign values to them.

Example 1

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

$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);

Example 2

This example is the same as example 1, but shows a different way of creating the array:

$ages['Peter'] = "32";

$ages['Quagmire'] = "30";

$ages['Joe'] = "34";

The ID keys can be used in a script:

<?php

$ages['Peter'] = "32";

$ages['Quagmire'] = "30";

$ages['Joe'] = "34";

echo "Peter is " . $ages['Peter'] . " years old.";

?>

The code above will output:

Peter is 32 years old.

**Multidimensional Arrays**

In a 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.

Example

In this example we create a multidimensional array, with automatically assigned ID keys:

$families = array

(

"Griffin"=>array

(

"Peter",

"Lois",

"Megan"

),

"Quagmire"=>array

(

"Glenn"

),

"Brown"=>array

(

"Cleveland",

"Loretta",

"Junior"

)

);

The array above would look like this if written to the output:

Array

(

[Griffin] => Array

(

[0] => Peter

[1] => Lois

[2] => Megan

)

[Quagmire] => Array

(

[0] => Glenn

)

[Brown] => Array

(

[0] => Cleveland

[1] => Loretta

[2] => Junior

)

)

Example 2

Lets try displaying a single value from the array above:

echo "Is " . $families['Griffin'][2] . " a part of the Griffin family?";

The code above will output:

Is Megan a part of the Griffin family?

Loops execute a block of code a specified number of times, or while a specified condition is true.