PHP Loops

In this tutorial you will learn how to repeat a series of actions using loops in PHP.

Different Types of Loops in PHP

Loops are used to execute the same block of code again and again, as long as a certain condition is met. The basic idea behind a loop is to automate the repetitive tasks within a program to save the time and effort. PHP supports four different types of loops.

- **while** — loops through a block of code as long as the condition specified evaluates to true.
- **do...while** — the block of code executed once and then condition is evaluated. If the condition is true the statement is repeated as long as the specified condition is true.
- **for** — loops through a block of code until the counter reaches a specified number.
- **foreach** — loops through a block of code for each element in an array.

You will also learn how to loop through the values of array using `foreach()` loop at the end of this chapter. The `foreach()` loop work specifically with arrays.

PHP while Loop

The `while` statement will loops through a block of code as long as the condition specified in the `while` statement evaluate to true.

```
while(condition){
    // Code to be executed
}
```

The example below define a loop that starts with `$i=1`. The loop will continue to run as long as `$i` is less than or equal to 3. The `$i` will increase by 1 each time the loop runs:

Example **Run this code »**

```
<?php
$i = 1;
while($i <= 3){
    $i++;
    echo "The number is " . $i . "<br>";
}
?>
```

PHP do...while Loop

The `do-while` loop is a variant of while loop, which evaluates the condition at the end of each loop iteration. With a `do-while` loop the block of code executed once, and then the condition is evaluated, if the condition is true, the statement is repeated as long as the specified condition evaluated to is true.

```
do{
    // Code to be executed
}
while(condition);
```

The following example define a loop that starts with `$i=1`. It will then increase `$i` with 1, and print the output. Then the condition is evaluated, and the loop will continue to run as long as `$i` is less than, or equal to 3.

Example **Run this code »**

```
<?php
$i = 1;
do{
    $i++;
    echo "The number is " . $i . "<br>";
}
while($i <= 3);
?>
```

Difference Between while and do...while Loop

The `while` loop differs from the `do-while` loop in one important way — with a `while` loop, the condition to be evaluated is tested at the beginning of each loop iteration, so if the conditional expression evaluates to false, the loop will never be executed.

With a `do-while` loop, on the other hand, the loop will always be executed once, even if the conditional expression is false, because the condition is evaluated at the end of the loop iteration rather than the beginning.

PHP for Loop

The `for` loop repeats a block of code as long as a certain condition is met. It is typically used to execute a block of code for certain number of times.

```
for(initialization; condition; increment){
    // Code to be executed
}
```

The parameters of `for` loop have following meanings:

- `initialization` — it is used to initialize the counter variables, and evaluated once unconditionally before the first execution of the body of the loop.
- `condition` — in the beginning of each iteration, condition is evaluated. If it evaluates to `true`, the loop continues and the nested statements are executed. If it evaluates to `false`, the execution of the loop ends.
- `increment` — it updates the loop counter with a new value. It is evaluate at the end of each iteration.

The example below defines a loop that starts with `$i=1`. The loop will continued until `$i` is less than, or equal to 3. The variable `$i` will increase by 1 each time the loop runs:

Example **Run this code »**

```
<?php
for($i=1; $i<=3; $i++){
    echo "The number is " . $i . "<br>";
}
?>
```

PHP foreach Loop

The `foreach` loop is used to iterate over arrays.

```
foreach($array as $value){
    // Code to be executed
}
```

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

Example **Run this code »**

```
<?php
$colors = array("Red", "Green", "Blue");

// Loop through colors array
foreach($colors as $value){
    echo $value . "<br>";
```

```
}
?>
```

There is one more syntax of `foreach` loop, which is extension of the first.

```
foreach($array as $key => $value){
    // Code to be executed
}
```

Example **Run this code »**

```
<?php
$superhero = array(
    "name" => "Peter Parker",
    "email" => "peterparker@mail.com",
    "age" => 18
);

// Loop through superhero array
foreach($superhero as $key => $value){
    echo $key . " : " . $value . "<br>";
}
?>
```