PHP - Break Statement

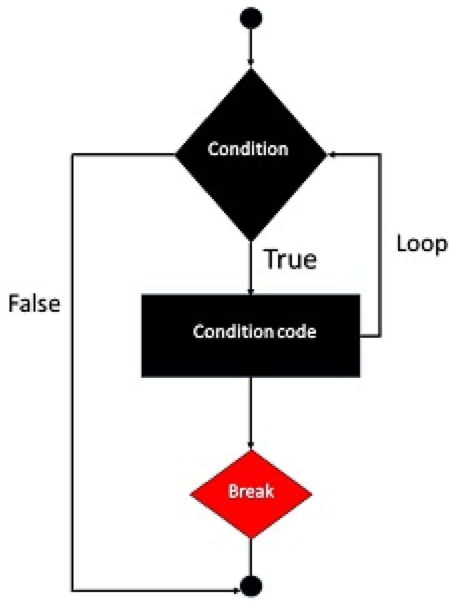
The **break** statement along with the **continue** statement in PHP are known as "loop control statements". Any type of loop (**for**, **while** or **do-while**) in PHP is designed to run for a certain number of iterations, as per the test condition used. The **break** statement inside the looping block takes the program flow outside the block, abandoning the rest of iterations that may be remaining.

The **break** statement is normally used conditionally. Otherwise, the loop will terminate without completing the first iteration itself.

The **syntax** of break statement is as follows −

```
while(expr){
   if (condition){
      break;
   }
}
```

The following **flowchart** explains how the **break** statement works −



Example

The following PHP code is a simple example of using **break** in a loop. The **while** loop is expected to perform ten iterations. However, a **break** statement inside the loop terminates it when the counter exceeds 3.

Open Compiler

```
<?php
   $i = 1;

   while ($i<=10){
      echo "Iteration No. $i \n";
      if ($i>=3){
         break;
      }
      $i++;
   }
?>
```

It will produce the following **output** −

```
Iteration No. 1
Iteration No. 2
Iteration No. 3
```

An optional numeric argument can be given in front of break keyword. It is especially useful in nested looping constructs. It tells how many nested enclosing structures are to be broken out of. The default value is 1, only the immediate enclosing structure is broken out of.

Example

The following example has three nested loops: a **for** loop inside which there is a **while** loop which in turn contains a **do-while** loop.

The innermost loop executes the **break.** The number "2" in front of it takes the control out of the current scope into the **for** loop instead of the immediate **while** loop.

Open Compiler

```
<?php
   for ($x=1; $x<=3; $x++){
      $y=1;
      while ($y<=3){
         $z=1;
         do {
            echo "x:$x y:$y z:$z \n";
            if ($z==2){
               break 2;
            }
            $z++;
         }
         while ($z<=3);
         $z=1;
         $y++;
      }
   }
?>
```

It will produce the following **output** −

```
x:1 y:1 z:1
x:1 y:1 z:2
x:2 y:1 z:1
x:2 y:1 z:2
x:3 y:1 z:1
x:3 y:1 z:2
```

Note that each time the value of "z" becomes 2, the program breaks out of the "y" loop. Hence, the value of "y" is always 1.