Schleifen: Immer wieder dasselbe tun

Bei einer Schleife werden Anweisungen immer wieder ausgeführt, solange die Bedingung wahr ist.

Dafür muss man eine Variable immer wieder ändern, solange bis eine Überprüfung nicht mehr zutrifft. Das ist dann das Ende.

Man unterscheidet verschiedene Arten von Schleifen:

- while
- for
- do-while
- for-each – wird bei arrays verwendet

Man nimmt meistens die Variable „$i“ als Zählervariable bei Schleifen, also als Basis.

Schleifen durchlaufen mit „while“:

Die einfachste Form einer Schleife in PHP ist die while-Konstruktion.
Bei einer While-Schleife werden Anweisungen ausgeführt, solange die Bedingung wahr ist. Ein Block wird in Abhängigkeit einer Bedingung mehrfach abgearbeitet.
Damit die Schleife nicht endlos (ENDLOSSCHLEIFE) läuft, sollte man im Anweisungsblock eine Veränderung der zu prüfenden Variable einfügen, z.B. $i ++.

Die Bedingungen werden in runde Klammern nach „while“ geschrieben. Die Definition erfolgt vor der „while-Schleife“ – hier z.B. $i=10. Die Anweisungen stehen in Mengenklammern.

```
while (Bedingung) {
        Anweisung_1;
        Anweisung_2;
        Anweisung_3;
        }
```

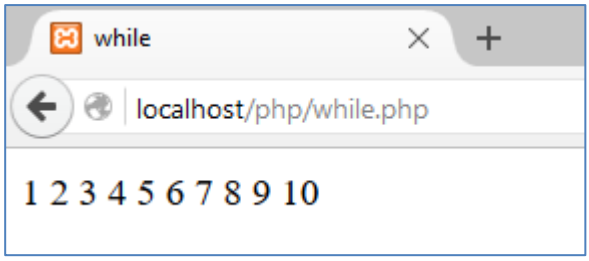
Solange die Bedingung den Wert „true“ besitzt, werden die Anweisungen in den geschweiften Klammern ausgeführt. Mindestens eine dieser Anweisungen muss also dafür sorgen, dass die Bedingung den Wert „false“ erhält, ansonsten wird die Schleife nie verlassen, und das Programm würde festhängen. Das Programm würde abstürzen. In diesem Fall kann es passieren, dass der Rechner für einige Zeit „einfriert“ und auf keine Eingabe mehr reagiert. Hier hilft in der Regel nur, abwarten. Denn irgendwann ist es dem Server genug und er beendet von sich aus die Verbindung. Notfalls kann man aber vorher den Browser beenden.

Beispiel: öffne eine neue php-Datei und speichere unter „while.php“
Innerhalb der Schleife wird jeweils der aktuelle Wert Von $anfang ausgegeben. Da dieser Wert mit $anfang++ bei jedem Durchlauf um 1 erhöht wird, ist $anfang irgendwann größer als $ende. Damit wird die Bedingung ($anfang<=$ende) false und die Schleife verlassen.

```
8   <body>
9   <?php
10  $anfang = 1;
11  $ende = 10;
12  while ($anfang <= $ende) {
13      echo $anfang, " ";
14      $anfang++;
15  }
16  ?>
17  </body>
```

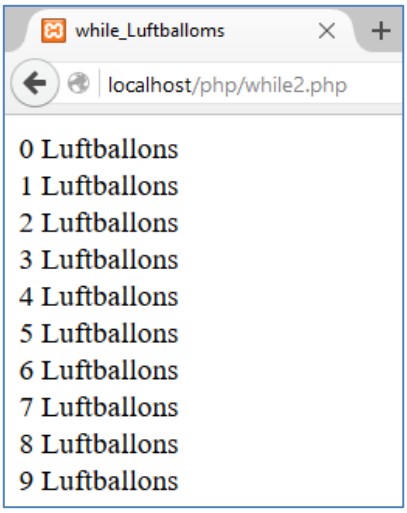
Durch den Beistrich und die Anführungszeichen mit einer Leertaste wird ein Abstand erzeugt und das Ergebnis sieht besser aus.

Ergebnis:

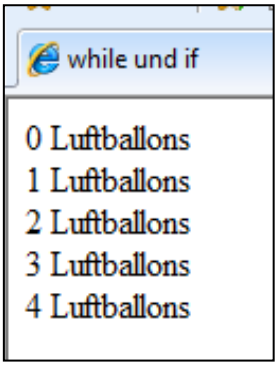


Beispiel: Luftballons: *(while2.php)*

```
8   <body>
9   <?php
10  $i = 0;
11  while ($i < 10)
12      {
13      echo "$i Luftballons <br>";
14      $i ++;
15      }
16  ?>
17  </body>
18  </html>
```



Erweiterung: *(while2_mit_if.php)*
Bearbeite die Datei „while2.php“ weiter und forme diese um, damit mit Hilfe von „break“ beim Wert 5 abgebrochen wird.
Tipp: verwende dazu eine if-Anweisung.
Speichern unter „while_2_mit_if.php“.



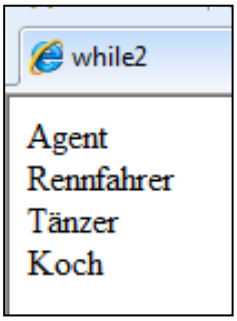
Beispiel Denksport:
Was passiert hier? Interpretiere die mögliche Lösung.

```
<?php
$zaehler = 0;
$ende = 4;
    while ($zaehler < $ende)
    {
    echo zaehler . "<br>";
    $zaehler++;
    }
?>
```

Beispiel: (*while3.php*)
Im folgenden Beispiel werden in der while-Schleife alle Werte eines **eindimensionalen Arrays** ausgegeben:

```
7
8   <body>
9   <?php
10  $beruf = array("Agent", "Rennfahrer", "Tänzer", "Koch");
11  $i=0;
12  while ($i <count ($beruf))
13  {
14      echo $beruf[$i] . "<br>";
15      $i++;
16  }
17
18  ?>
```

Ergebnis:



Arrays lassen sich in einer foreach-Schleife noch einfacher ausgeben.

Schleifen durchlaufen mit „for“:

Eine while-Schleife wird solange durchlaufen, bis ihre Bedingung nicht mehr erfüllt ist. Damit lassen sich in ihr Abläufe wiederholen, die umgangssprachlich mit „so lange, bis“ formuliert werden. Z.B. „Lies die Datensätze aus der Datenbank, bis das Ende erreicht ist.“ Man weiß aber vorher nicht, wie oft eine Schleife durchlaufen werden muss.
Daneben gibt es aber auch immer wieder Fälle, in denen von vornherein klar ist, wie oft bestimmte Anweisungen wiederholt werden sollen. **Hier kennt PHP die for-Schleife**, die ähnlich wie while arbeitet und aber ihre Stärken für alle irgendwie abzählbaren Bedingungen besitzt.

Im Unterschied zu einer while-Schleife wird nicht innerhalb der Schleife dafür gesorgt, dass die Bedingung der Schleife einmal „false“ erreicht, sondern die **notwendige Steueranweisung steht bereits zu Beginn** der Schleife.

```
for (Initialisierung; Bedingung; Veränderung) {
        beliebige Anweisungen
}
```

In der Schleife müssen (in runder Klammer) drei Werte eingegeben werden, die durch **Semikolon getrennt** sind:

- Zuerst die **Initialisierung (Start)** der Variablen, die geprüft werden soll. Sie wird genau einmal beim Start ausgeführt.
- anschließend die **Bedingung**, auf die geprüft wird. Solange diese „true“ ist, wird die Schleife durchlaufen, z.B. $a <=10.
- Und schließlich ein Ausdruck, der angibt, wie die zu prüfende Variable bei jedem Schleifendurchlauf **verändert** werden soll,

in der geschweiften Klammer steht dann das, was innerhalb der Schleife passiert.

Die Schleife läuft so lange, bis die Prüfung der Bedingung falsch ergibt.
Die einzelnen Ausdrücke der For-Anweisung müssen durch **Strichpunkte** getrennt werden. Die Anweisungen, die mittels der Schleife wiederholt ausgeführt werden sollen, müssen innerhalb einer geschwungenen Klammer stehen.

Besteht die Schleife allerdings nur aus einer Anweisung, lässt sich diese auch ohne Mengenklammer in einer Zeile angeben:

```
for (Initialisierung; Bedingung; Veränderung) Anweisung;
```

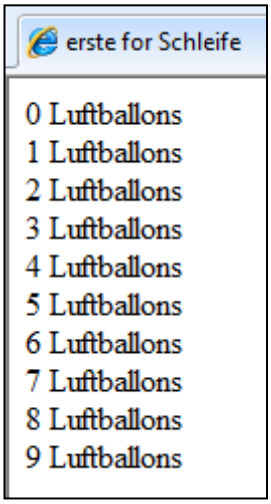
Beispiel:

```
<?php
  for ($a =1; $a <= 10; $a++) echo $a, " ";
?>
```

Beispiel: Aufblasen von Luftballons

(Übung: for1.php)

```
 8  <body>
 9  <?php
10  for ( $i = 0; $i < 10; $i ++) {
11      echo $i . " Luftballons <br>";
12  }
13
14  ?>
```



Der **Befehl break** bewirkt, dass die Ausführung der Schleife sofort beendet wird, egal, ob die Bedingung noch weiterhin wahr wäre oder nicht. Break darf nicht direkt in einer Schleife stehen, da sie sonst nie ausgeführt würde, sondern nur in einem If-Befehl.

Mit der **Anweisung continue** wird der aktuelle Durchlauf der Schleife übersprungen und mit dem nächsthöherem Zähler fortgefahren. Ebenso wie break darf auch continue nur in einem if-Befehl und nicht direkt auftreten.

Quellen:

Lynn Beighley und Michael Morrison in: PHP & MySQL von Kopf bis Fuß;
Verlag =O´Reilly, 2009

Johann-Christian Hanke in: PHP und MySQL; bhv VERLAG; 2010

Gisbert Damaschke in: PHP & MySQL, Wiley-Vch Verlag, 2015; S.109-113