3 Grundlegendes: Datentypen, Kontrollstrukturen

Da Sie schon andere Programmiersprachen kennen, hier nur eine kurze Übersicht über die üblichen Datentypen und Kontrollstrukturen.

3.1 Datentypen

Versuchen Sie die Datentypen zu verstehen. Nicht genannt sind die Datentypen resource für eine Ressource wie ein Bild und object.

```php
<?php
$bool = true;
$integer = 42;
$double = 1.235;
$string ="Text";
$array = array("Frankfurt", "Berlin", "Stuttgart");
echo "true ist gleich " . gettype($bool);
echo "<br>42 ist ein " .  gettype($integer);
echo "<br>1.256 ist ein " .  gettype($double);
echo "<br> Text ist ein " .  gettype($string);
echo "<br> Frankfurt, Berlin, ... ist ein " .  gettype($array);
echo "<br>eine undefinierte Variable hat den Wert " .  gettype($neu);
?>
```

3.2 Kontrollstrukturen

3.2.1 Vergleichsoperatoren

Operator	Name	Bedeutung
==	gleich	$a == $b ist dann TRUE, wenn $a und $b gleich sind Beispiel $a = 4; $b=4.0 $a == $b; Ergebnis: TRUE
!= oder <>	ungleich	$a != $b ergibt TRUE, wenn $a und b$ ungleich sind
===	identisch	$a === $b ergibt TRUE, wenn $a und $b gleich und vom selben Datentyp sind. Beispiel $a = 4; $b=4.0 $a === $b; Ergebnis: FALSE
!==	nicht identisch	$a !== $b ergibt TRUE, wenn $a und $b ungleich oder nicht vom selben Datentyp sind
<	kleiner	
>	größer	
<=	kleiner gleich	
>=	größer gleich	
< = >	Spaceship Operator	ein Vergleich mit diesem Operator ergibt 1, falls der erste Wert größer ist, -1, falls der zweite Wert größer ist, 0, falls beide Werte übereinstimmen
?	Ternärer Operator	`$preis = 1.98;` `echo ($preis < 1 ? "das ist billig" : "Langsam wird es teuer") ." ";` Schreibabkürzung für Verzweigungen, enthält drei Operanden Bedingung $preis<1 ? trifft zu : trifft nicht zu nach :
??	Koaleszenzoperator	Verschmilzt isset mit dem Ternären Operator `echo ($preis ?? „Nicht vorhanden“)` gibt aus „Nicht vorhanden“

Verknüpfen von Bedingungen mit folgenden Operatoren

Operator	Name	Bedeutung
AND &&	Logisches Und	Alle Bedingungen müssen zutreffen
||	Logisches Oder	Nur eine von mehreren Bedingungen muss zutreffen

3.2.2 Einfache if-Anweisung

```
if (Bedingung) {
    Anweisungsblock
}
```

- Als Bedingung ist ein logische Ausdruck anzugeben, der einen der beiden Zustände TRUE oder FALSE zurückliefert.
- Liefert die Bedingung TRUE zurück, werden die Anweisungen ausgeführt, ist die Bedingung FALSE, werden die Anweisungen ignoriert.

3.2.3 if-Anweisung mit else Zweig

```
if (Bedingung) {
            Anweisungsblock 1
}
else  {
      Anweisungsblock 2
}
```

3.2.4 Existenz einer Variablen

Man kann die Existenz einer Variablen mithilfe der Funktion isset() prüfen. Sie können auf diese Weise z. B. feststellen, ob bestimmte Werte aus einem Formular gesendet werden.
Die Funktion isset() liefert einen Wahrheitswert, daher wird sie meist innerhalb einer Verzweigung eingesetzt.

→ Öffnen Sie die Datei 08_isset.php und ergänzen Sie im Quellcode die Erläuterungen, warum in dem jeweiligen Fall die Variable existiert oder nicht.

3.2.5 Typ prüfen

Der Typ einer Variablen kann folgendermaßen geprüft werden:

- is_int()
- is_float()
- is_string()
- is_numeric()
- is_bool()

→ Datei 09_typ_pruefen.php öffnen und verstehen.

3.2.6 Schleifen

Schleifen sind werden benutzt, wenn sich innerhalb eines Programms einzelne Anweisungen oder Blöcke wiederholen. Es gibt die for-Schleife, die while Schleife und die do-while Schleife.

3.2.6.1 for Schleife

Eine for-Schleife wird verwendet, wenn die Anzahl der Wiederholungen bekannt ist oder diese sich eindeutig im Verlauf des Programms vor der Schleife ergibt (Zählschleife).

```
<?php
   for ($i=1; $i<=5; $i++)
   {
      echo "Zeile $i<br>";
   }
?>
```

3.2.6.2 while-Schleife

Man verwendet eine while-Schleife oder die do-while-Schleife, wenn die Anzahl der Wiederholungen nicht bekannt ist und diese sich nicht eindeutig im Verlauf des Programms vor der Schleife ergibt. Die Wiederholung oder der Abbruch der Schleife ergibt sich erst zur Laufzeit des Programms (bedingungsgesteuerte Schleife).

```
<?php
 srand((double)microtime()*1000000); //Initialisierung, nicht unbedingt notwendig
   $summe = 0;

   while ($summe < 25)
   {
      $zufallszahl = rand(1,6); //zufällige Zahlen zwischen 1 und 6
      $summe = $summe + $zufallszahl;
      echo "Zahl $zufallszahl, Summe $summe<br>";
   }

?>
```

3.2.6.3 do-while-Schleife

Die do-while-Schleife arbeitet wie die while-Schleife, es gibt aber einen wichtigen Unterschied: Die Prüfung für die Wiederholung wird erst am Ende der Schleife durchgeführt. Die Schleife wird also mindestens einmal ausgeführt.

```
<?php
srand((double)microtime()*1000000);
$summe = 0;
do
{
    $zufallszahl = rand(1,6);
    $summe = $summe + $zufallszahl;
    echo "Zahl $zufallszahl, Summe $summe<br>";
}
while ($summe < 25);
?>
```

3.2.6.4 break

Mit Hilfe der Anweisung break kann eine Schleife vorzeitig beendet werden.
In diesem Programm wird eine zusätzlicher Zähler verwendet, der zunächst auf 0 gesetzt wird. Innerhalb der Schleife wird dieser jeweils um 1 erhöht. Wir dabei die Zahl 6 erreicht, bricht die Schleife unmittelbar ab, auch wenn die Summe noch kleiner 25 ist.

```
<?php
   srand((double)microtime()*1000000);
   $summe = 0;
   $zaehler = 0;

   while ($summe < 25)
   {
      $zufallszahl = rand(1,6);
      $summe = $summe + $zufallszahl;
      $zaehler = $zaehler + 1;
      echo "Nr. $zaehler, Zahl $zufallszahl,";
      echo " Summe $summe<br>";
      if ($zaehler >= 6) break;       // Sonderfall
   }
?>
```