4. Funktionen

Aus Gründen
1. der Übersichtlichkeit
2. der Wiederverwendbarkeit von Code
3. der geringeren Schreibarbeit
4. der geringeren Fehleranfälligkeit

von Programmen teilt man ein Programm in verschiedene Anweisungen auf. In den Programmiersprachen werden verschiedene Wörter für diese Anweisungsblöcke verwandt, die wir der Einfachheit halber alle synonym verwenden wollen: Prozeduren, Funktionen, Anweisungen.

4.1 Vergleich zur Karol

In Karol haben wir Anweisungen bereits kennengelernt:

Beispiel:

```
anweisung umdrehen
   linksdrehen
   linksdrehen
*anweisung
```

Der Name der Anweisung, hier `"umdrehen"` kann beliebig gewählt werden. Ist eine Anweisung definiert, kann im Hauptprogramm die Funktion einfach durch Verwendung des Namen aufgerufen werden.

4.2 Anweisungen in PHP

In PHP ändert sich an der Definition von Funktionen nichts Grundlegendes. Anstelle von Anweisung verwendet man das Schlüsselwort `"function".` Der Name ist wiederum beliebig, jedoch schließt der Name immer mit runden Klammern ab: `"()"`. Anstelle eines Ende-Zeichens werden die Befehle einfach in geschweifte Klammern gesetzt.

`Beispiel:`

```
function Einladung(){
   echo "Hallo Anton<br>";
   echo "Hiermit lade ich dich ein am 10.10.2004<br>";
   echo "zu meiner Geburtstagsfeier.<br>";
   echo "Viele liebe Gruesse<br>";
   echo "Dein Hans Dampf";
}
```

4.3 Wo stehen die Funktionen

Wir befinden uns stets in HTML-Dokumenten. Die Funktionen definieren wir am besten am Ende, also nach dem </HTML>-Tag. Beachte, dass auch alle Funktionen zu PHP gehören und daher innerhalb von PHP-Tags stehen müssen.

Der Aufruf der Funktionen erfolgt innerhalb des <BODY>-Tags. Auch dabei handelt es sich um PHP-Befehle, also benötigen wir auch hier die PHP-Tags.

Beispiel, der graue Text ist HTML, der orange sind die PHP-Tags, das blaue der Aufruf und der Name der Funktion:

```
<html><head><title>Meine Erste Funktion</title></head>
<body>
<p>Hier steht ganz normaler HTML-Text<br>
In der naechsten Zeile beginnt der Funktionsaufruf<br>
<?php Einladung(); ?>
Durch das Ende des PHP-Tags ist dies wieder normaler HTML-
Code.
</body>
</html>

<?php
function Einladung(){
   echo "Hallo Anton<br>";
   echo "Hiermit lade ich dich ein am 10.10.2004<br>";
   echo "zu meiner Geburtstagsfeier.<br>";
   echo "Viele liebe Gruesse<br>";
   echo "Dein Hans Dampf";
}
?>
```

4.4 Parameterübergabe

Die Funktion "Einladung()" ist nicht besonders spannend. Sinnvoll lassen sich Funktionen oft nur anwenden, wenn Variablen existieren. Schauen wir uns auch hierzu ein Beispiel an, das sich an obigem orientiert. Zu Beginn werden in einem PHP-Tag zwei Variablen festgelegt, $Name und $Datum. Die Funktion "Einladung()" verändern wir auch in der Art, dass sie nur mit zwei Variablen aufgerufen werden kann. Die verlangten Variablen stehen in den runden Klammern. In der Funktion dürfen die Variablen andere Namen haben. Sie bekommen beim Aufruf im Hauptprogramm auf jeden Fall die richtigen übergeben, man spricht von Parameter-Übergabe.

```
<?php
$Name="Anton"; $Datum="10.10.2004";
?>

<html><head><title>Meine Erste Funktion</title></head>
<body>
<p>Hier steht ganz normaler HTML-Text<br>
In der naechsten Zeile beginnt der Funktionsaufruf<br>
<?php Einladung($Name, $Datum); ?>
Durch das Ende des PHP-Tags ist dies wieder normaler HTML-
Code.
</body>
</html>

<?php
function Einladung($Var1, $Var2){
   echo "Hallo $Var1<br>";
   echo "Hiermit lade ich dich ein am $Var2<br>";
   echo "zu meiner Geburtstagsfeier.<br>";
   echo "Viele liebe Gruesse<br>";
   echo "Dein Hans Dampf";
}
?>
```

Es ist nicht möglich, eine Funktion in der oben beschriebenen Art ohne Variablenübergabe auszuführen. Variablen, die im Hauptprogramm deklariert wurden, sind in der Funktion nämlich unbekannt.

4.5 Funktionen mit Rückgabewert

Funktionen können auch einen Rückgabewert haben, also ein Ergebnis. Gekennzeichnet werden diese Ergebnisse durch das Schlüsselwort `return`. Der Vorteil einer solchen Implementierung liegt u.a. daran, dass die Funktion selbst wie eine Variable verwendet werden kann.

Es macht keinen Sinn, eine Funktion mit Ergebnis einfach so aufzurufen, ohne dass das Ergebnis einer Variable oder einem Befehl zugeordnet ist.

Beispiel:

```
$Ergebnis=BerechneFlaeche(10,20);
echo $Ergebnis;
// oder

echo BerechneFlaeche(10,20);

fucntion BerechneFlaeche($Laenge, $Breite){

 return $Laenge * $Breite;

}
```