

6. Funktionen

Funktionen sind Programmteile, die zur Lösung bestimmter Aufgaben vorgesehen sind. Der PHP-Skript-Programmierer kann Funktionen selbst definieren und so immer wiederkehrende Aufgaben von diesen Funktionen erledigen lassen, ohne hunderte von Zeilen Programmcode erstellen zu müssen. Dadurch lässt sich eine bessere Strukturierung des Programmcodes erreichen, die wiederum eine bessere Lesbarkeit bewirkt.

Eine Funktion ist also - mit anderen Worten - eine Bündelung von Befehlen. Man unterscheidet im Allgemeinen drei Typen von Funktionen:

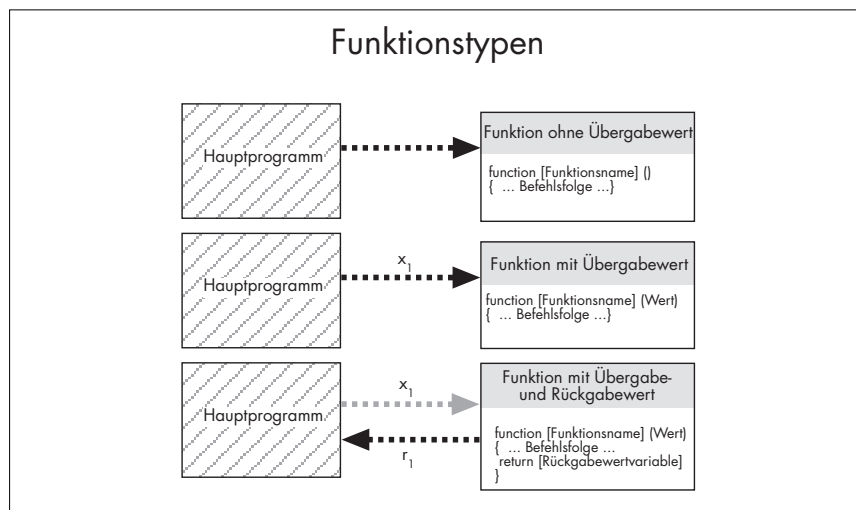


Abb. 6.1 Funktionstypen

6.1 Funktionen ohne Übergabewert

Eine Funktion beginnt immer mit dem Schlüsselwort »function«, darauf folgt der Funktionsname und der Operator »()«. Dem Funktionsnamen mit dem Operator »()« folgt eine öffnende geschweifte Klammer. Die Funktion selber definiert sich über den folgenden Block. Letztes Zeichen ist eine schließende geschweifte Klammer. Ausgeführt werden Funktionen, indem man sie über ih-

ren Namen aufruft. Der Aufruf an sich kann überall im PHP-Skript platziert werden, wo er sinnvoll erscheint. Er muss jedoch nach der Funktionsdefinition erfolgen.

Bei einer Funktion ohne Übergabewert wird die Funktion nur mit dem Funktionsnamen, gefolgt von einer öffnenden und schließenden runden Klammer, aufgerufen. Die Befehlsfolge, die in der Funktion definiert ist, wird dann abgearbeitet. Die Vergabe der benutzerdefinierten Funktionsnamen erfolgt nach den Regeln der Variablennamenvergabe (siehe Kap. 3.1). Allgemeine Syntax:

```

Definition:    function Funktionsname() {
                // Anweisungen
            }

Aufruf:       Funktionsname();
    
```

listing_0601.php

```

<?php
function textausgabe() {
    echo "Dies ist ein Beispieltext!";
    echo "Ausgegeben durch eine Funktion. <br>";
}

// 2 Funktionsaufrufe
textausgabe();
textausgabe();
?>
    
```

Funktion wird erst ausgeführt, wenn diese aufgerufen wird

Abb. 6.2 Funktionen ohne Übergabewert

Bei jedem Funktionsaufruf wird die Befehlsfolge in der Funktion komplett abgearbeitet. Nachdem das erfolgt ist, wird das Programm fortgeführt. Im obigen Beispiel wird eine Funktion »textausgabe()« definiert. Die Funktion enthält zwei echo-Anweisungen, die einen Text ausgeben. Dann erfolgt zweimal der Funktionsaufruf durch »textausgabe()«.

6.2 Funktionen mit Übergabewert

Häufig unterscheiden sich Funktionen nur durch ihre Anfangswerte. D.h. den Text, den wir in der Funktion »textausgabe()« im Kapitel 6.1 ausgegeben haben, könnte man auch durch eine oder mehrere Variablen ersetzen. Die Zeichenketten erhalten dann diese Variablen durch die Übergabewerte. Die allgemeine Syntax sieht wie folgt aus:

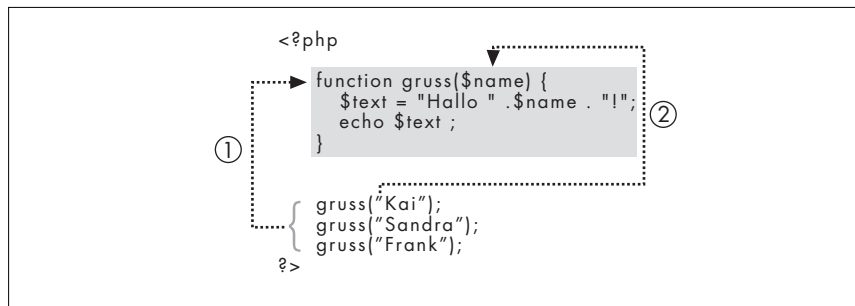
```
function name( Argument1 [, Argument2] ... [, Argument n] ){
    ... Befehlsfolge
}
```

Die Argumente bei der Funktionsdefinition werden auch als »formale« Parameter oder kurz Parameter bezeichnet.

Beim Aufruf der Funktion sollte die Anzahl der übergebenen Werte (Übergabewerte) mit der Anzahl der in der Funktion definierten Parameter übereinstimmen. Sollte beim Aufruf einer Funktion mehr als ein Wert übergeben werden, so sind diese durch Kommata zu trennen.

```
name(Wert1 [, Wert2], ... [, Wert n]);
```

Die beim Aufruf übergebenen Werte werden auch »aktuelle« Parameter genannt.



listing_0602.php

Abb. 6.3 Funktionen mit Übergabewert

Im Beispiel Abbildung 6.3 wird die Funktion »gruss()« definiert, die einen Parameter besitzt, den wir hier mit »\$name« bezeichnet haben. Dieser erhält beim Aufruf der Funktion mit »gruss()« den Wert »Kai«, »Sandra« und »Frank«. Die Funktion wird also Folgendes auf dem Bildschirm ausgeben: »Hallo Kai! Hallo Sandra! Hallo Frank!«.

Wenn man sicher gehen möchte, dass die Funktion existiert, bevor man sie aufruft, kann man dies mit der PHP-Funktion »function_exists()« prüfen.

```
<?php
function text_ausgabe( $x ) {
    echo "Der uebergebene Text lautet: ". $x;
}
// *** Funktionsaufruf ***
if ( function_exists( "text_ausgabe" ) ) {
    text_ausgabe("Hallo!");
}
?>
```

listing_0603.php

Im obigen Beispiel findet nun der Funktionsaufruf nur statt, wenn die Funktion »text_ausgabe« existiert. Die Funktion »function_exists()« gibt entweder ein »true« oder ein »false« zurück.

Es gibt auch Fälle, in denen eine Funktion noch nicht existiert. Definieren wir eine Funktion innerhalb einer Funktion, so ist diese solange unbekannt, bis die Funktion aufgerufen wird, die diese Funktion definiert.

listing_0604.php

```
<?php
function ausgabe( $x ) {
    echo "Die uebergebene Zahl lautet: ". $x;
    function quadrat($z){
        $q = $z*$z;
        echo "Das Quadrat von $z ist $q";
    }
    quadrat( $x );
}
// *** Funktionsaufruf ***
ausgabe(4) ;
?>
```

In diesem Beispiel muss die Funktion »ausgabe()« zuerst einmal abgearbeitet werden, bevor die Funktion »quadrat()« definiert ist. Hierbei kann der Funktionsaufruf dann auch aus einer Funktion getätigt werden.

Die Art und Weise, wie wir hier die Variablen übergeben haben, nennt man »call-by-value«. Es wird nur der Wert einer Variablen übergeben, nicht die Variable selbst. Die Variable im Hauptprogramm ändert sich nicht. Ebenso existiert eine Methode, wobei nur eine Referenz übergeben wird. Welche Vor- und Nachteile das hat erfahren wir in Kapitel 6.3.1.

6.3 Funktionen mit Rückgabewerten

Eine Funktion kann auch einen Wert mit Hilfe des return-Befehls zurückgeben. Dies ist besonders nützlich für Berechnungen und für solche Anwendungen, in denen man die Funktion in einen Ausdruck einsetzen möchte. Ein klassisches Beispiel stellt das Quadrieren von Zahlen da.

Im folgenden Beispiel rufen wir die Funktion mit einer Wertübergabe auf. Der Parameter »Zahl« erhält den Wert »6«. Zusätzlich wird einer Variablen »\$x_qua« der Rückgabewert zugewiesen. Der Rückgabewert wird mit dem Schlüsselwort »return« innerhalb der Funktion »quadrieren()« zurückgegeben, sodass dann schließlich »36« ausgegeben wird.

```

<?php
function quadrieren($zahl) {
    $y = 0;
    $y = $zahl * $zahl;
    return $y; // gibt das berechnete Quadrat zurück
}
② → $x_qua = quadrieren(6); // berechnet das Quadrat von 6
    ①
echo "Das Quadrat von 6 ist: ".$x_qua;
    // Wert erhält x_qua, also x_qua = 36
?>

```

listing_0605.php

Abb. 6.4 Funktionen mit Übergabe- und Rückgabewert

Grundsätzlich kann eine Funktion nur einen Rückgabewert zurückgeben. Soll eine Funktion mehr als einen Rückgabewert zurückgeben, so kann dies mit Hilfe eines Arrays geschehen. Ein Beispiel aus der Praxis wäre die Aufschlüsselung eines Datumwertes in Jahr, Monat, Tag (Funktion »substr()«, siehe Kap. 10.3):

```

<?php
$datum = "20020923";
function datum_extr($dat){
    $d["Jahr"] = substr($dat,0,4);
    $d["Monat"] = substr($dat,4,2);
    $d["Tag"] = substr($dat,6,2);
    return $d;
}
$d_einzel = datum_extr($datum);
echo "Tag: ".$d_einzel["Tag"];
echo "Tag: ".$d_einzel["Monat"];
echo "Tag: ".$d_einzel["Jahr"];
?>

```

listing_0606.php

6.3.1 Übergabe durch Referenz

Wenn Sie Werte einer Funktion übergeben, behalten die Variablen ihre ursprünglichen Werte bei. Sollen diese Werte jedoch ebenfalls verändert werden, so müssen Sie die Referenz (Adresse) auf diese Variable der Funktion übergeben. Man nennt dieses Verfahren »call-by-reference«. Es ermöglicht dem Programmierer über eine selbst definierte Funktion auf die Variablenwerte des Hauptprogramms Einfluss zu nehmen. Dies sieht folgendermaßen aus:

listing_0607.php

```

<?php
function ausgabe( &$p ) {
    echo "Die &uuml;bergene Zahl lautet: ". $p; // Ausgabe: x=2
    $p = 4;
}
$x = 2;
ausgabe($x) ; // *** Funktionsaufruf ***
echo "x hat jetzt den Wert: $x"; // Ausgabe : 4
?>

```

In diesem Beispiel rufen wir die Funktion »ausgabe()« mit dem Übergabewert »\$x« auf. Wir übergeben hier jedoch nicht den Wert von »\$x« also »2«, sondern den Zeiger »Pointer« auf die Variable »\$x«. Das drücken wir mit einem kaufmännischem »&« im Parameter der Funktion »ausgabe(&\$p)« aus. Jede neue Wertzuweisung der Variablen »\$p« innerhalb der Funktion wirkt sich auch auf den Variablenwert außerhalb der Funktion aus. Jeder Wert, der dem Pointer »\$p« übergeben wird, ist gleichzeitig der neue Wert für »\$x«. Hierbei erfolgt die Zuordnung der Werte entsprechend der Reihenfolge, in der sie der Funktion übergeben werden. Wird mehr als ein Pointer übergeben, sieht das wie folgt aus:

listing_0608.php

```

<?php
function ausgabe( &$p,&$q){
    echo "Die &uuml;bergene Zahlen lauten: $p und $q";
    $p=4;
    $q = 99;
}
$x = 2;
$y = 9;
ausgabe($x,$y);
echo "<br>x hat jetzt den Wert: $x";
echo "<br>y hat jetzt den Wert: $y";
?>

```

Doch was für Vorteile bringt diese Art der Übergabe? Werden viele Parameter mit dem »call-by-value«-Verfahren übergeben, müssen die Werte vom PHP-Interpreter kopiert werden, damit sie auch außerhalb einer Funktion verwendet werden können. Bei der Adressenübergabe »call-by-reference« braucht man keinen Wert kopieren und daraus resultiert der Geschwindigkeitsvorteil dieser Methode.

6.3.2 Standardwerte bei Funktionen

Bei der Definition von Funktionen mit Übergabeparametern können Standardwerte definiert werden, falls diese Funktion ohne Parameterübergabe aufgerufen wird.

```
<?php
function ausgabe( $x=99 ) {
    echo "<br>Die &uuml;bergebene Zahl lautet: ". $x;
}
// *** Funktionsaufruf ***
ausgabe(5) ; // Hierbei erhaelt $x den Wert 5
ausgabe(); // Hierbei erhaelt $x den Wert 99
?>
```

listing_0609.php

Wird also die Funktion mit einem Parameter aufgerufen, so wird die Wertzuweisung »99« ignoriert. Wird die Funktion ohne Parameter aufgerufen, wird der Variablen »\$x« der Standardwert »99« zugewiesen. Auch wenn mehr als ein Parameter definiert ist, funktioniert das Prinzip des Standardwertes.

```
<?php
function ausgabe( $x=99, $y=9 ) {
    echo "<br>Die x &uuml;bergebene Zahl lautet: ". $x;
    echo "<br>Die y &uuml;bergebene Zahl lautet: ". $y;
}
// *** Funktionsaufruf ***
ausgabe(5); // Hierbei erhaelt $x den Wert 5, $y 9
ausgabe(5,10); // Hierbei erhaelt $x den Wert 5, $y 10
ausgabe(); // Hierbei erhaelt $x den Wert 99,$y 9
?>
```

listing_0610.php

6.4 Globale und lokale Variablen

Die Gültigkeitsbereiche von Variablen in PHP sind auf die Umgebung reduziert, in der sie deklariert wurden. Unter globalen Variablen versteht man Variablen, die im ganzen Programm gültig sind. Globale Variablen werden außerhalb von Funktionen deklariert und sind im Gegensatz zu anderen Programmiersprachen in PHP auch nur außerhalb von Funktionen bekannt. Soll eine globale Variable auch innerhalb von Funktionen bekannt sein, muss die Anweisung »global« der Variablen vorangestellt werden.

```
<?php
$x = "Wasser";
$y = "Feuer";
```

listing_0611.php

```
function ausgabe(){
    global $x, $y;
    echo $x." ".$y;
}
?>
```

Durch das Schlüsselwort »global« werden global definierte Variablen auch für Funktionen bekannt gemacht und können ausgelesen und verändert werden. Der veränderte Wert wird nach dem Abarbeiten des Skriptes übernommen. Es gibt aber noch eine andere Möglichkeit, auf globale Variablen zuzugreifen und zwar mit dem in PHP vordefinierten Array »\$GLOBALS«. Dieses Array ist sowohl außerhalb als auch innerhalb von Funktionen gültig.

listing_0612.php

```
<?php
    $x = 34.6;
    $y = 20;
    function ausgabe(){
        echo $GLOBALS ["x"]."<br>"; // Ausgabe ergibt: 34.6
        echo $GLOBALS ["y"]."<br>"; // Ausgabe ergibt: 20
    }
?>
```

Bei dem Array »\$GLOBALS« handelt es sich um ein assoziatives Array (siehe Kap. 7), dessen Schlüssel der Variablenname ist. Im Allgemeinen sollte man versuchen, globale Variablen zu vermeiden. Für nahezu alle Probleme lassen sich lokale Lösungen finden. Programmteile, die von globalen Variablen abhängig sind, lassen sich kaum wieder verwenden und sind schwerer zu analysieren.

Im Gegensatz zu globalen Variablen werden lokale Variablen innerhalb einer Funktion deklariert. Sie sind nur während der Ausführungszeit der Funktion existent und gültig. Ist die Funktion abgearbeitet, sind die Variablen nicht mehr vorhanden. Es gibt eine Möglichkeit, die Werte dieser Variablen auch nach Abarbeitung der Funktion zu erhalten, indem die Anweisung »static« der Variablen innerhalb der Funktion vorangestellt wird.

listing_0613.php

```
<?php
    function zaehler(){
        static $x = 0;
        echo "x ist $x";
        $x ++;
    }
    zaehler();
    zaehler();
?>
```

Der statischen Variablen »\$x« wird der Anfangswert »0« zugeordnet. Der Wert wird ausgegeben und anschließend um eins erhöht. Beim zweiten Aufruf wird der Wert der statischen Variablen nicht wieder auf Null gesetzt, sondern nur ausgegeben und dann wieder um eins erhöht.

6.5 Modularisierung mit include() / require()

Manchmal ist es notwendig, eine selbstdefinierte Funktion in mehreren PHP-Skripten zu verwenden. Um diese Funktion nicht in allen PHP-Skripten implementieren zu müssen, wo sie aufgerufen wird, kann sie in einer separaten Datei ausgelagert werden. Mit Hilfe der Anweisungen »include/require« kann die Funktion in das entsprechende Skript eingebunden werden. Ist eine Änderung der Funktion notwendig, braucht nur die ausgelagerte Datei verändert zu werden und man erspart sich viel Arbeit die man hätte, wenn dieselbe Funktion in mehreren PHP-Skripten definiert worden wäre.

Dabei wird zwischen »include()« und »require()« unterschieden. Mit »require()« wird der Inhalt einer Datei noch während des Parsens, bevor die erste Skriptcode-Zeile ausgeführt wird, eingebunden.

Die »include()«-Funktion arbeitet etwas anders. Die Einbindung erfolgt erst während der Laufzeit. Damit eröffnet sich die Möglichkeit, das Einfügen über Laufzeitvariablen zu kontrollieren und so ein Skript dynamisch zu erweitern. Werden allerdings innerhalb einer ausgelagerten Datei Funktionen deklariert, wird ein erneutes Einbinden in das Skript zu Fehlermeldungen führen. Deswegen sollte man bei der Auslagerung von Funktionen das Einbinden mit »require()« vorziehen. Um sicherzustellen, dass wirklich nur einmal eine Datei eingebunden wird, wurden in PHP die Funktionen »include_once()« und »require_once()« implementiert.

Beim Auslagern von Funktionen »require« verwenden

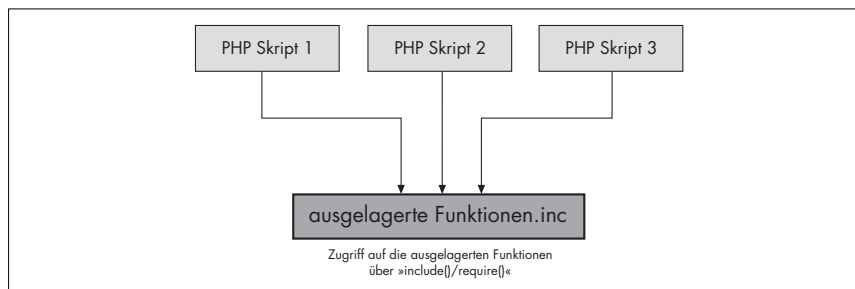


Abb. 6.5 Mit »include()/require()« ausgelagerte Dateien

Zuerst wird die auszulagernde Funktion in einer separaten Datei, beispielsweise

se mit dem Namen »quadrat.inc.«, gespeichert:

```
listing_0614a.php <?php
                  // *** Berechnung des Quadrates einer Zahl ***
                  function quadratzahl ( $x ) {
                    $qz = $x * $x ;
                    return $qz;
                  }
                  ?>
```

Wird diese Funktion als Teil einer Funktionsbibliothek verwendet, darf keine Anweisung außerhalb der in ihr definierten Funktion auftreten, da diese dann sofort nach dem Einbinden ausgeführt werden würde. Man benötigt nun noch ein PHP-Skript, welches diese Datei einbindet und ausführt.

```
listing_0614b.php <?php
                  require_once( "quadrat.inc" );
                  $ergebnis = quadratzahl ( 4 );
                  echo "Das Quadrat von 4 ist $ergebnis ! "; // $ergebnis = 16
                  ?>
```

Die Endung ».inc« ist nicht vorgeschrieben, aber eine Konvention, die die meisten Programmierer verwenden. Häufig wird auch die Endung ».inc.php« verwendet, wenn die ausgelagerte Datei PHP-Skripte enthält. Theoretisch sind alle Endungen möglich, weil der PHP-Interpreter die Datei einbinden muss.

include()

Die Funktion »include()« sollte verwendet werden, wenn bestimmte Teile eines PHP-Skriptes ausgelagert werden sollen. Taucht z.B. ein größerer HTML-Block auf, so ist es sinnvoll, diesen auszulagern und mit Hilfe von »include()« einzubinden. Ebenso sollten sicherheitsrelevante Teile eines Skriptes immer ausgelagert und mit »include()« eingebunden werden, denn wenn der PHP-Interpreter einmal nicht arbeitet, wird das Skript als Klartext im Browser angezeigt, die ausgelagerten »include()«-Dateien jedoch nicht. Die ausgelagerte Datei wird immer an der Stelle im PHP-Skript eingebunden, an der die Anweisung »include()« auftaucht. So können »include()«-Dateien auch mit Hilfe von Kontrollstrukturen eingebunden werden:

```
listing_0615.php <?php
                  $bedingung = false;
                  if ($bedingung == true){
                    include ("formatierung_1.inc");
                  }
                  }
```

```

else{
    include ("formatierung_2.inc");
}
?>

```

require()

Üblicherweise werden Funktionen mit »require()« am Anfang eines Skriptes einmalig eingebunden und dann verwendet. Hierbei ist ein wiederholtes Aufrufen weder nötig noch sinnvoll und würde zu Fehlermeldungen führen, da eine Funktion immer nur einmal definiert werden.

6.6 Sitemanagement mit Include-Dateien

Das folgende Beispiel, das wir Ihnen vorstellen möchten, verdeutlicht die Bedeutung der »include()«-Funktion in Bezug auf das Auslagern von HTML-Teilen, die für das Layout einer Webseite verantwortlich sind. Nehmen wir einmal an, unsere Seite besteht aus einer Tabelle mit den Abschnitten: Auswahlmenü, Content-Teil und Fußzeile. Das Auswahlmenü ist in der include-Datei »c_kopf.inc« untergebracht, der Hauptteil, in dem sich der Seiteninhalt befindet, ist in der Datei »c_news.php« abgelegt und die abschließende Fußzeile für das Aktualisierungsdatum und das Copyright in der Datei »c_fuss.inc«.

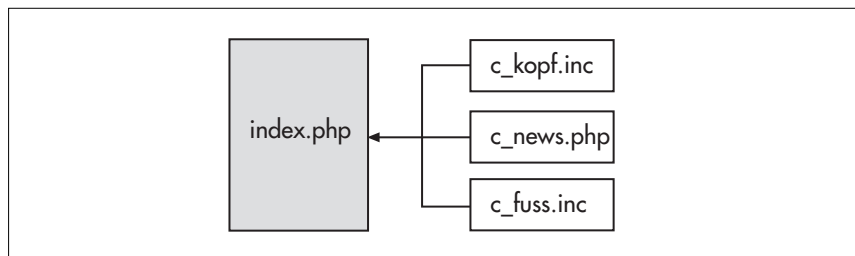


Abb. 6.6 Sitelayout mit Include-Dateien

Diese drei Include-Dateien werden in der Datei »index.php« zusammengefasst. Da jeder dieser Abschnitte in einer Include-Datei ausgelagert ist, sind spätere Änderungen im Layout einfach zu realisieren. In der Abbildung 6.7 sieht man deutlich, dass die Seiten »index.php«, »links.php« und »kontakt.php« auf die gleichen Include-Dateien zugreifen. Wenn man im Menü oder in der Fußzeile Änderungen durchführen möchte, muss man nur die Dateien »c_kopf.inc« oder »c_fuss.inc« verändern.


```

        <td>&nbsp;</td>
        <td>&nbsp;</td>
    </tr>
    <tr>
        <td bgcolor="#CCCCCC">&nbsp;</td>
        <td width="800" height="500" bgcolor="#EAF FE8">&nbsp;</td>
        <td bgcolor="#CCCCCC">&nbsp;</td>
    </tr>
    <tr bgcolor="#CCCCCC">
        <td>&nbsp;</td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
    </tr>
</table>
</body>
</html>

```

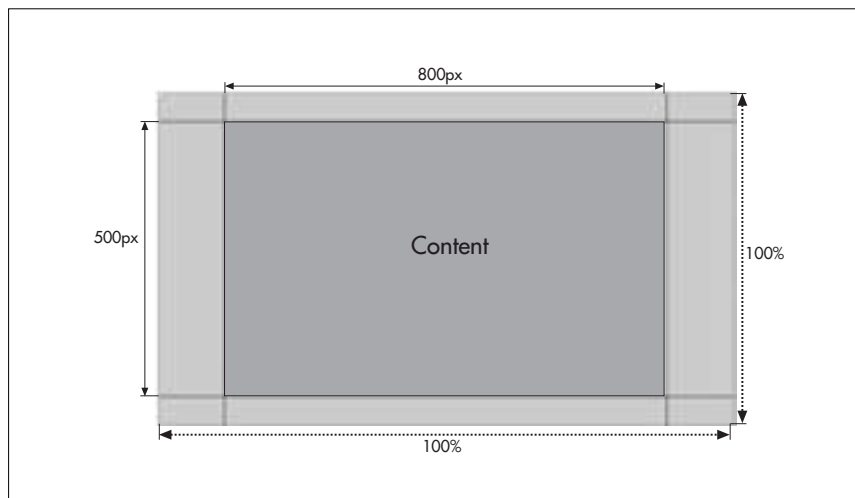


Abb. 6.8 Layout für eine flexible Größenanpassung

Hierbei gilt zu beachten, dass zu diesem Zeitpunkt die mittlere Tabellenzelle noch kleiner wird, wenn das Browserfenster die Größe von 800x500 Pixel unterschreitet. Das ändert sich aber, wenn die Zelle durch die Grafiken auf der Größe gehalten wird.

Im nächsten Schritt wird eine 800x500 Pixel große Tabelle mit drei Zeilen erzeugt, die dann in die 800x500 Pixel große Tabellenzelle eingebettet wird. Diese drei Zeilen enthalten Menü, Inhalt und Fußzeile.

```

<table width="800" height="500" border="0">
    <tr>
        <td height="100">&nbsp;</td>
    </tr>

```

```

<tr>
  <td height="345"><p>&nbsp;</p>
</td>
</tr>
<tr>
  <td>&nbsp;</td>
</tr>
</table>

```

Wenn man nun die »kleine Tabelle« in das zuvor erstellte Layout eingefügt hat, sieht das Seiten-Layout wie folgt aus:

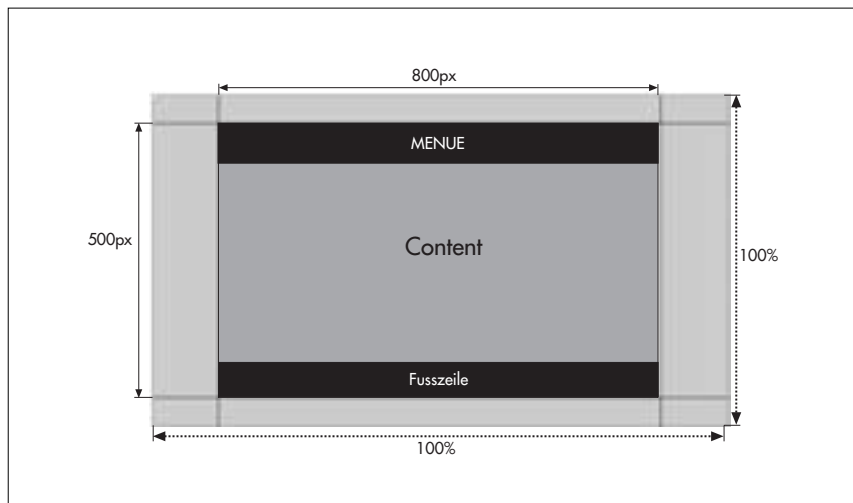


Abb. 6.9 Tabellenaufbau für eine dynamische Webseite

Das dazugehörige Listing:

listing_0617.php

```

<html>
<head>
<title>Layout</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body bgcolor="#FFFFFF">
<table width="100%" height="90%" border="0" align="center"
  cellpadding="1" cellspacing="0">
<tr bgcolor="#CCCCCC">
  <td>&nbsp;</td>
  <td>&nbsp;</td>
  <td>&nbsp;</td>
</tr>
<tr>
  <td bgcolor="#CCCCCC">&nbsp;</td>
  <td bgcolor="#CCCCCC" width="800" height="500" bgcolor="#EAF8E8">

```

```

<table width="800" height="500" border="0">
  <tr><td height="100">Menue</td></tr>
  <tr><td>Content</td></tr>
  <tr><td height="50">Fusszeile</td></tr>
</table>
</td>
<td bgcolor="#CCCCCC">&nbsp;</td>
</tr>
<tr bgcolor="#CCCCCC">
  <td>&nbsp;</td>
  <td>&nbsp;</td>
  <td>&nbsp;</td>
</tr>
</table>
</body>
</html>

```

Als Letztes müssen die Dateien »c_kopf.inc«, »c_news.php« und »c_fuss.inc« erzeugt werden, die in der Datei »index.php« zusammengefasst werden. In der Datei »index.php« soll dann Folgendes stehen:

```

<?php
include("c_kopf.inc");
echo "Home";
include ("c_fuss.inc");
?>

```

listing_0618.php

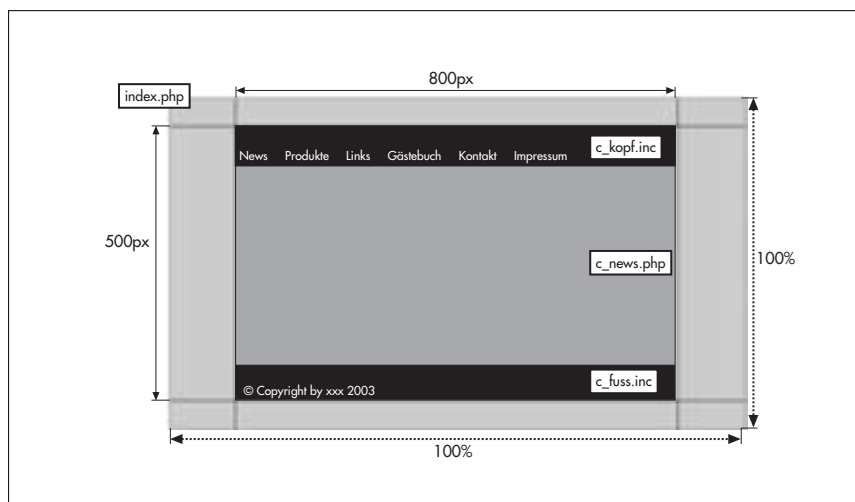


Abb. 6.10 Sitemanagement mit Include-Dateien

Schauen wir uns zuerst einmal das Listing der auszulagernden Datei »c_kopf.

Stelle durchgeführt wird, da sie sich ja hinter einem »<td>« befindet und die Tabellenzelle noch nicht abgeschlossen ist. Die Erklärung ist ganz einfach. Der Content-Teil soll ja in einer Tabellenzelle eingegliedert werden und diese Einbettung erfolgt nach dem Öffnen einer Tabellenzelle. Das Schließen der Zelle erfolgt in der Datei »c_fuss.inc«. Der folgende Teil des Listings wird in der Datei »c_fuss.inc« ausgelagert.

```

</td>
</tr>
<tr>
<td colspan="6" bgcolor="#333333">
<font color="#FFFFFF" size="1" face="Verdana, Arial, Helvetica, sans-serif">
  &copy; Copyright by PHP-interaktiv 2003 </font>
</td>
</tr>
</table></td>
<td bgcolor="#CCCCCC">&nbsp;</td>
</tr>
<tr bgcolor="#CCCCCC">
  <td>&nbsp;</td>
  <td>&nbsp;</td>
  <td>&nbsp;</td>
</tr>
</table>
</body>
</html>

```

listing_0620.php

Dieses Listing wird in zwei Teile getrennt und in den Dateien »c_kopf.inc« und »c_fuss.inc« abgespeichert. Die Trennung erfolgt dort, wo der Content-Teil eingebettet werden soll, denn bis auf den Content-Teil bleibt der übrige HTML-Code in jedem Menüpunkt immer gleich. Die beiden ausgelagerten Dateien werden dann in den einzelnen Themen-Dateien wieder eingebunden. In unserem Fall waren das »News«, »Produkte«, »Links«, »Gästebuch«, »Kontakt« und »Impressum«. Beispielsweise sehen die Dateien »gaestebuch.php« und »kontakt.php« so aus:

```

<?php
  include ("c_kopf.inc");
  include ("gaestebuch.php");
  include ("c_fuss.inc");
?>

```

listing_0621.php

und:

```

<?php
  include ("c_kopf.inc");

```

```
include ("kontakt.php");  
include ("c_fuss.inc");  
?>
```

Wir haben unser Ziel erreicht. Zwei Dateien, die unser Layout bestimmen und eine, die für den Content zuständig ist. Jetzt kann man sagen, dass ist doch nichts Besonderes, dasselbe Ergebnis erreicht man ebenso mit drei HTML-Frames. Das ist richtig, doch hat unsere PHP-Methode den Vorteil, dass sie für Suchmaschinen günstiger zu indizieren ist, als eine *geframete* Seite und dieser Vorteil ist nicht unerheblich.

Übungen und Aufgaben

1. Schreiben Sie ein Skript, welches eine Funktion beinhaltet, die einen Übergabeparameter akzeptiert. Der Übergabeparameter kann Zahlen zwischen 1 und 7 annehmen. Die Funktion gibt dann die entsprechenden Wochentagsnamen zurück, d.h. »Montag« für den Wert 1, »Dienstag« für den Wert 2 usw.!
2. Schreiben Sie ein Skript, welches eine Funktion enthält, die die Quadratwurzel einer Zahl berechnet. Als Übergabewerte werden nur positive Zahlen akzeptiert, ansonsten wird eine Fehlermeldung herausgegeben!
3. Sie haben in Kapitel 6 gelernt, wie man dynamisch eine Tabelle erstellt. Schreiben Sie eine Funktion für das Erstellen einer Tabelle. Wählen Sie hierbei die Übergabeparameter so, dass die Funktion möglichst vielseitig eingesetzt werden kann.