

1 Oberfläche zum Ausführen der PHP-Dateien

PHP-Skripte werden vom Webserver ausgeführt. Wenn lokal am Rechner PHP-Dateien erstellt und getestet werden, dann ist es erforderlich, sie im Internet-Explorer mit ihrer lokalen Adresse aufzurufen, also zum Beispiel:

`http://localhost/test.php`

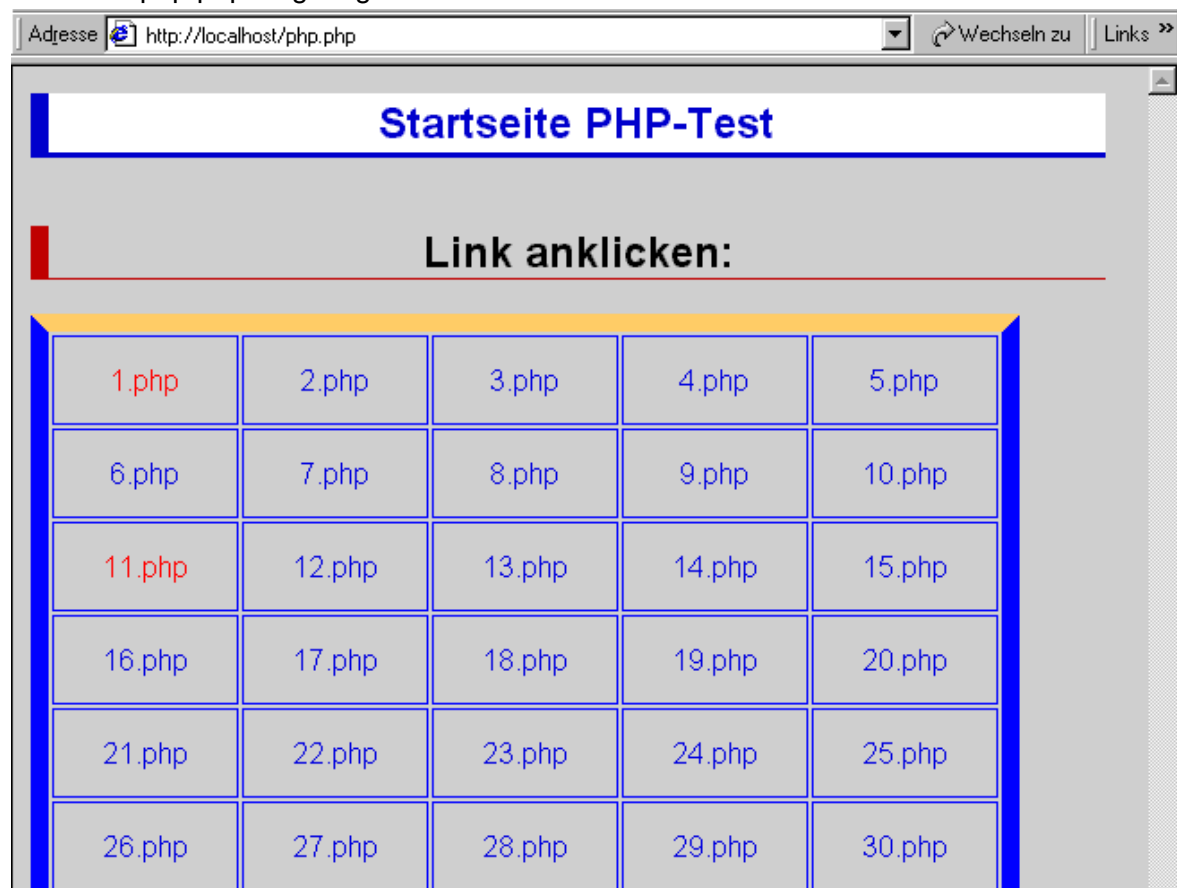
(dazu muss sich außerdem die Datei im htdocs-Ordner befinden)

Keinesfalls können die Dateien durch einen Doppelklick über Arbeitsplatz oder Windows Explorer aufgerufen werden, denn in diesem Fall wird nur der HTML-Anteil "verstanden", der Webserver (hier: Apache) "merkt" ja dann gar nichts davon, dass diese Datei aktiviert wurde.

Es ist zu umständlich, jedes Mal die Adresse komplett einzugeben. Und weil sich außerdem die Dateien (als default-Einstellung) im htdocs-Ordner befinden müssen, oder man muss den Pfad zusätzlich eintippen – aus diesem Gründen ist es einfacher, wenn wir uns zunächst eine Oberfläche schaffen, über die wir durch Anklicken des Links die Dateien öffnen. Die Datei **php.php** enthält eine Tabelle mit Links, z. B. **1.php**. Durch Anklicken wird die Datei **1.php** im Ordner **C:\Apache2\htdocs\phptest** aufgerufen (bzw. bei einer XAMPP-Installation im Ordner C:\XAMPP\htdocs).

Wir können die Dateien, die übungshalber erstellt werden sollen, nun nacheinander mit den Dateinamen 1.php, 2.php, 3.php usw. im Ordner **C:\Apache2\htdocs\phptest** erstellen und speichern (mit dem HTML-Editor, PHP-Coder oder welchen Editor Sie bevorzugen), dann von der einmal gestarteten Datei **php.php** aus die einzelnen Übungen über den Link starten und testen.

Die Datei `php.php` zeigt folgenden Aufbau:



Die Datei php.php enthält das HTML-Grundgerüst, den Link zur CSS-Datei **standard.css** (die z. B. die Hover-Effekte beim Darüberstreichen mit der Maus schafft), die Tabelle und die Links.

php.php

```
<html>
<head>
<title>PHP-Testdateien</title>
<link rel="stylesheet" href="C:/Apache2/htdocs/phptest/standard.css"
type="text/css">
</head>
<body>
<h1>Startseite PHP-Test</h1>
<br>
<h2>Link anklicken:</h2>
<table bordercolor="#0000FF" border="1" >
<tr>
<td width="100" height="50"><a href="phptest/1.php">1.php</a></td>
<td width="100"><a href="phptest/2.php">2.php</a> </td>
<td width="100"><a href="phptest/3.php">3.php</a> </td>
<td width="100"><a href="phptest/4.php">4.php</a> </td>
<td width="100"><a href="phptest/5.php">5.php</a> </td>
</tr>
<tr>
.....usw für die anderen Tabellenzeilen.....
</tr></table>
</body>
</html>
```

2 PHP in HTML-Dateien

PHP-Sprachelemente werden in die HTML-Datei eingebettet. Sie eröffnen PHP mit

```
<?php
```

Nun folgen die PHP-Befehle. Zur Übergabe an HTML schließen Sie wieder mit

```
?>
```

2.1 Das Hallo-Welt-Skript

Aufgabe: Erstellen Sie die erste Datei zum Testen. Sie enthält den für den ersten Schritt mit einer Programmiersprache klassischen "Hallo-Welt" –Test. Sie macht nichts anderes, als den Text "Hallo Welt" auf den Bildschirm zu bringen:

1.php

```
<html>
<!-- 1.php -->
<head>
<link rel="stylesheet" href="standard.css" type="text/css">
<title></title>
</head>
<body>
<h1>Das Hallo-Welt-Skript</h1>
<p>
<?php
echo "Hallo Welt"
?>
</p>
<a href="http://localhost/php.php">zurück zur Index-Seite</a>
</body>
</html>
```



3 Kommentare

Es ist sinnvoll, im Quellcode mit Kommentaren zu arbeiten. Das Programm wird dadurch lesbarer. Vor allem, wenn man sich erst nach einer Zeit wieder den Code ansieht, versteht man ihn viel besser, wenn man mit Kommentaren gearbeitet hat. Für andere, die ihren Code lesen sollen, gilt dies natürlich erst recht.

Einzeilige Kommentare werden mit zwei // (Schrägstrichen) oder einer Raute gemacht:

```
<?php
echo "Hallo Welt"           //Ausgabe des Hallo-Welt-Standardtextes
$wert=3+4                   #der Variablen $wert wird 3 + 4 zugewiesen.
?>
```

Mehrzeilige Kommentare werden von /* und */ eingeschlossen.

```
$wert=3+4                   /*der Variablen $wert wird das Ergebnis der
Berechnung von 3 + 4 zugewiesen.*/
```

4 die Funktion phpinfo()

phpinfo zeigt Ihnen Infos über die Konfiguration von PHP an. Durch Aufruf dieser Funktion kann man sich davon vergewissern, welche Einstellungen gelten und welche Zusatzmodule ggf. geladen worden sind.

Aufgabe: Die zweite Datei soll die in PHP integrierte Funktion **phpinfo()** ausführen.

2.php

```
<html>
<!-- 2.php -->
<head>
<link rel="stylesheet" href="standard.css" type="text/css">
<title></title>
</head>
<body>
<h1>PHP-Info</h1>
<p>
<?php
phpinfo();
?>
</p>
<a href="http://localhost/php.php">zurück zur Index-Seite</a>
</body>
</html>
```

Sie zeigt (hier: Ausschnitt) die verschiedenen Einstellungen, die wichtig für das Funktionieren später benutzter Funktionen in PHP sind.

System	Windows 95/98 4.10
Build Date	Apr 30 2001
Server API	CGI
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\apache\php\php.ini
ZEND_DEBUG	disabled
Thread Safety	enabled

5 Variablen und Berechnungen in PHP

Variablen sind ein sehr wichtiger Bestandteil jeder Programmiersprache. Sie dienen dazu, Werte zwischenspeichern, die später wieder benötigt werden. In PHP gilt generell die Regel, dass Variablen mit einem \$-Zeichen beginnen. PHP unterscheidet zwischen Groß- und Kleinschreibung. Sonderzeichen (außer dem Unterstrich) dürfen für Variablennamen in PHP nicht verwendet werden.

Es ist nicht erforderlich, Variablen vor Gebrauch zu deklarieren. Den Datentyp (Integer für Ganzzahlen, Double für Dezimalzahlen, String für Text (muss innerhalb von Anführungsstrichen stehen) und Boolean (logisch, kann den Wert true oder false annehmen) wird von PHP automatisch erkannt.

Einer Variable wird der Wert durch ein = Zeichen zugewiesen.

`$zahl = 45` Die Variable \$zahl erhält den Wert 45

`$text = "Wolfgang"` Die Variable \$text bekommt den Inhalt **"Wolfgang"** zugewiesen

Achtung: beim Vergleich muss ein doppeltes == Zeichen verwendet werden! Das einfache Gleichheitszeichen weist nur einen Wert zu.

Generell gelten für Berechnungen die Rechenzeichen, die auch in Anwendungsprogrammen wie z. B. Excel verwendet werden:

Verwenden Sie folgende Rechenzeichen:

+	addieren
-	subtrahieren
*	multiplizieren
/	dividieren

Außerdem gilt eine abgekürzte Schreibweise für das Erhöhen bzw. Vermindern um 1, wie es auch in einigen anderen Sprachen eingesetzt wird.

Statt des Ausdrucks: | schreibt man kürzer:

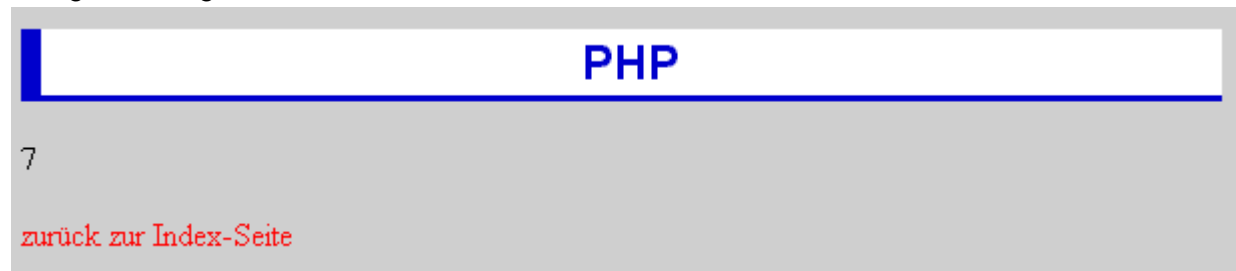
<code>\$z=\$z +1</code>	<code>\$++</code>
<code>\$x=\$x-1</code>	<code>\$x --</code>
<code>\$y=\$y+2</code>	<code>\$y+=2</code>
<code>\$w=\$w-5</code>	<code>\$w-=5</code>

Aufgabe: Erstellen Sie die nächste Datei. Mit ihr wird getestet, wie eine Variable verwendet wird und eine einfache Rechnung ausgeführt wird:

3.php

```
<html>
<!-- 3.php -->
<head>
<link rel="stylesheet" href="standard.css" type="text/css">
<title></title>
</head>
<body>
<h1>PHP</h1>
<p>
<?php
$c=3+4;
echo $c;
?>
</p>
<a href="http://localhost/php.php">zurück zur Index-Seite</a>
</body>
</html>
```

Der Variable mit dem Namen **\$c** wird das Ergebnis der Rechnung 3+4 zugewiesen. Danach erfolgt die Ausgabe **echo \$c** . Sie sehen auf dem Bildschirm:



Aufgabe: Erstellen Sie die nächste Übungsdatei.

4.php

```
<html>
<!-- 4.php -->
<head>
<link rel="stylesheet" href="standard.css" type="text/css">
<title></title>
</head>
<body>
<h1>PHP</h1>
<p>
<?php
$a=5;
$b=8;
$c=$a+$b;
echo $c;
?>
</p>
<a href="http://localhost/php.php">zurück zur Index-Seite</a>
</body>
</html>
```

Hier wird zunächst den Variablen **\$a** und **\$b** eine Zahl zugewiesen. Dann erfolgt die Addition und die Zuweisung des Ergebnisses in die Variable **\$c**.

Auf dem Bildschirm müsste zu sehen sein:

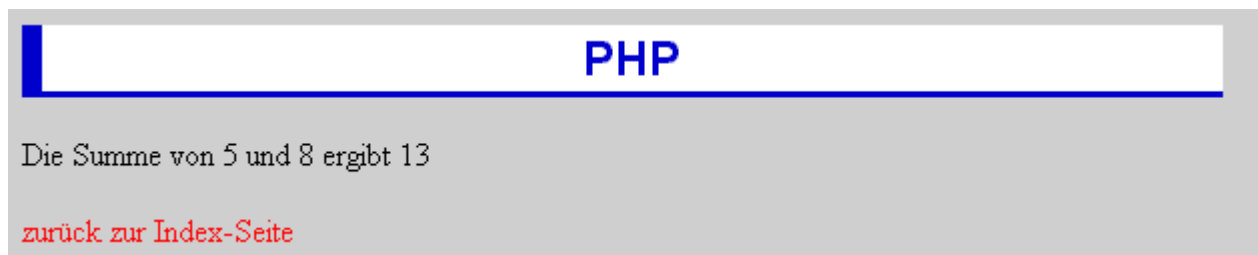


Aufgabe: Erstellen Sie die Datei **5.php**. Der Unterschied ist hier nur, dass in der Ausgabezeile die Verwendung von Variablen und Text zu sehen ist. Innerhalb der Anführungszeichen nach **echo** wird alles ausgegeben, sei es einfacher Text, Variablen oder auch HTML.

5.php

```
<html>
<!-- 5.php -->
<head>
<link rel="stylesheet" href="standard.css" type="text/css">
<title></title>
</head>
<body>
<h1>PHP</h1>
<p>
<?php
$a=5;
$b=8;
$c=$a+$b;
echo "Die Summe von $a und $b ergibt $c";
?>
</p>
<a href="http://localhost/php.php">zurück zur Index-Seite</a>
</body>
</html>
```

Auf dem Monitor sehen Sie:



Aufgabe: Ändern Sie an der folgenden Datei nur die **echo-Zeile** und speichern Sie mit dem Namen **6.php**. Hier soll gezeigt werden, wie innerhalb der Ausgabe auch HTML-Tags verwendet werden können:

6.php

```
...
echo "Die <i>Summe</i> von <b>$a</b> und <b>$b</b> <u>ergibt</u>
<b>$c</b>";
...
```

Als Ausgabe sehen Sie:



Aufgabe: Erstellen Sie nun die Datei **7.php**.

```
<html>
<!-- 7.php -->
<head>
<link rel="stylesheet" href="standard.css" type="text/css">
<title></title>
</head>
<body>
<h1>Verkettung von Strings</h1>
<p>
<?php
$text1="Der Anfang";
$text2="und die Fortsetzung";
echo "$text1 $text2";
?>
</p>
<a href="http://localhost/php.php">zurück zur Index-Seite</a>
</body>
</html>
```

Die Ausgabe ist folgende:

Der Anfang und die Fortsetzung

An diesem Beispiel ist zunächst noch nichts Neues. In der Fortsetzung sehen Sie, worauf das Ganze hinauslaufen soll.

Aufgabe: Erstellen Sie Datei **8.php**. Hier soll die Verkettung von Zeichen durch den Punkt-Operator gezeigt werden. Wozu in Javascript das Plus-Zeichen dient, wird in PHP der Punkt verwendet.

Verändern Sie an der Datei **8.php** gegenüber der Datei 7.php nur die echo-Zeile:

```
echo ($text1 . $text2);
```

Die Ausgabe ist dieselbe wie bei der zuvor erstellten Datei. Bei diesem Beispiel ist der Punkt als Verkettungsoperator tatsächlich gar nicht nötig, aber für die Ausgabe aus Funktionen wird dieses Verfahren benötigt. Außerdem wird es, wie im nächsten Beispiel zu sehen, verwendet, um mit einer verkürzten Schreibweise nach und nach Werte zu verketteten, indem sie aneinandergesetzt werden.

9.php

```
<html>
<!-- 9.php -->
<head>
<link rel="stylesheet" href="standard.css" type="text/css">
<title></title>
</head>
<body>
<h1>Verkettung von Strings</h1>
<p>
<?php
$text="Der Anfang ";
$text.="und die Fortsetzung";
echo ($text);
?>
</p>
<a href="http://localhost/php.php">zurück zur Index-Seite</a>
</body>
</html>
```

Die Ausgabe wird wieder dieselbe sein wie bei den beiden Dateien vorher. –

Hier geschieht folgendes; Zuerst in der Variablen `$text` nur "der Anfang". In der nächsten Zeile `$text.="und die Fortsetzung";` wird dem bisherigen Inhalt von `$text` über den Punkt etwas hinzugefügt, nämlich der zweite Teil des Textes. Dann erfolgt wieder die Ausgabe des Textes.

Dieses Verfahren wird häufig verwendet, um nach und nach den Inhalt einer String-Variablen mit immer mehr Inhalt zu füllen.

5.1 Arrays

Im Unterschied zu den bisher verwendeten Variablen sind **Arrays** mehrdimensional. Das bedeutet, dass man in einem Array mehr als einen Wert speichern kann. Über einen Index (die Zahl in eckigen Klammern) sind die Werte aufzurufen.

Aufgabe: Erstellen Sie die folgende Datei:

10.php

```
<html>
<!-- 10.php -->
<head>
<link rel="stylesheet" href="standard.css" type="text/css">
<title></title>
</head>
<body>
<h1>Arrays in PHP</h1>
<p>
<?php
$vorname[0]="Herbert";
$vorname[1]="Johannes";
$vorname[2]="Dieter";
$vorname[3]="Petra";
$vorname[4]="Beate";
$vorname[5]="Gerd";
$vorname[6]="Karl-Heinz";
// den vierten Namen ausgeben
echo $vorname[3];
?>
</p>
<a href="http://localhost/php.php">zurück zur Index-Seite</a>
</body>
</html>
```

Da die Zählung im Array mit 0 beginnt, ist der vierte Vorname der, der dem Array mit `$vorname[3]="Petra";` zugewiesen wurde.

Der Monitor zeigt:



Aufgabe: Arrays können auch in einer abgekürzten Form verwendet werden. Das soll das folgende Beispiel zeigen:

11.php

```
<html>
<!-- 11.php -->
<head>
<title>Arrays</title>
<link rel="stylesheet" href="standard.css" type="text/css">
</head>
<body>
<body>
<h1>Array in Kurzform</h1>

<p>
<?php
$name=array("Schmitz","Meyer","Müller","Paulsen","Demorgitas","Lesotra",
"Lohmann");
// Den fünften Namen ausgeben
echo $name[4];
?>
</p>
<a href="http://localhost/php.php">zurück zur Index-Seite</a>

</body>
</html>
```

Der Code **11.php** führt zu dieser Ausgabe:



Aufgabe: Erstellen Sie jetzt die Datei **12.php**. Dem Array **\$tag** werden die einzelnen Wochentage zugewiesen. Die in PHP integrierte `date()`-Funktion mit dem Zusatz "w" liefert den aktuellen Tag als Zahl, beginnend mit Sonntag=0. Das entspricht der Indizierung, die wir über den Array vornehmen:

12.php

```
<html>
<!-- 12.php -->
<head>
<title>Arrays</title>
<link rel="stylesheet" href="standard.css" type="text/css">
</head>
<body>
<body>
<h1>Tag mit Array ermitteln</h1>
<p>
<?php
```

```
$tag=array("Sonntag","Montag","Dienstag","Mittwoch","Donnerstag","Freitag",
"Samstag");
$tagzahl=date("w"); // Tag ermitteln
echo "Heute ist $tag[$tagzahl].";
?>
</p>
<a href="http://localhost/php.php">zurück zur Index-Seite</a>
</body>
</html>
```

Die Ausgabe muss den aktuellen Tag zeigen (Donnerstag also nur, wenn tatsächlich Donnerstag ist)

Tag mit Array ermitteln

Heute ist Donnerstag.

[zurück zur Index-Seite](#)

Oft ist es erforderlich, zu ermitteln, wie viel Werte ein Array enthält. Das gelingt mit der Funktion `count()`. Erstellen Sie dazu das folgende Beispiel:

13.php

```
<html>
<!-- 13.php -->
<head>
<!-- 13.php -->
<title>Arrays</title>
<link rel="stylesheet" href="standard.css" type="text/css">
</head>
<body>
<h1>Elemente im Array mit count() zählen</h1>

<p>
<?php
$tag=array("Sonntag","Montag","Dienstag","Mittwoch","Donnerstag","Freitag",
"Samstag");
$tagzahl=date("w"); // Tag ermitteln
echo "Heute ist $tag[$tagzahl].<br>";
$elementzahl=count($tag);
echo "Die Woche hat $elementzahl Tage.";
?>
</p>
<a href="http://localhost/php.php">zurück zur Index-Seite</a>
</body>
</html>
```

Als Ausgabe sehen Sie:

Elemente im Array mit count() zählen

Heute ist Samstag.
Die Woche hat 7 Tage.

6 Formulare

Wenn Eingaben vom Benutzer verwendet werden sollen, dann werden dazu HTML-Formulare verwendet. Die Verwendung der Werte erfolgt entweder in der Datei selber oder in einer anderen Datei, je nach dem, was in der `<form action=..... >`-Zeile angegeben wird. Wenn Sie in dieser Zeile den Dateinamen der Datei verwenden, in der Sie gerade sind, dann werden die Eingaben innerhalb der Datei selbst verwendet.

```
<form action = "14.php" method = "post">
```

Hier wird innerhalb der Datei 14.php der Wert an die Datei selbst übergeben.

Das einfachste Formular-Element ist ein Textfeld, wie Sie es in der folgenden Zeile sehen:

```
Wie heißt du? <input type = "text" name="vorname">
```

Hier wird eine Eingabe in das Textfeld erwartet.

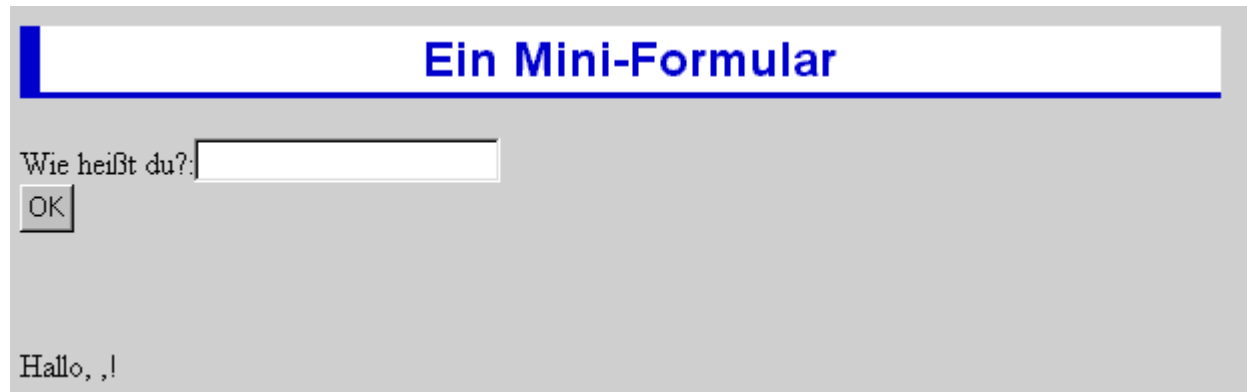
Wie heißt du?

In HTML wird der Inhalt in die Variable **vorname** übergeben. Dies wird automatisch in PHP zur Variablen **\$vorname** und kann weiter verarbeitet werden.

Aufgabe: Erstellen Sie die Datei **14.php**

```
<html>
<!-- 14.php -->
<link rel="stylesheet" href="standard.css" type="text/css">
<head>
<title>Formular</title>
</head>
<body>
<h1>Ein Mini-Formular</h1>
<form action = "14.php" method = "post">
Wie heißt du?:<input type = "text" name="vorname"></input><br>
<input type="submit" name="OK" value="OK"></input>
</form>
<br><br>
<?php
echo "Hallo, <b>$vorname</b>,!";
?>
<a href="http://localhost/php.php">zurück zur Index-Seite</a>
</body></html>
```

Wenn Sie die Datei aufrufen, dann sehen Sie zunächst:



Ein Mini-Formular

Wie heißt du?:

OK

Hallo, ,!

Diese Übungsdatei ist, wie Sie sehen, mit einem Schönheitsfehler behaftet, dass nämlich unten schon **Hallo, ,!** steht, bevor oben im Formulareil etwas eingegeben wurde. Das liegt einerseits daran, dass die Übergabe der Variablen von HTML an PHP in der Datei selber erfolgt, und andererseits daran, dass wir noch keine if-Unterscheidung zur Verfügung haben, um das zu verhindern.

Sobald ein Vorname eingegeben wurde und **OK** angeklickt wird, sehen Sie unten z. B.:

Hallo, **Hermann-Josef**,!

7 If-Verzweigungen

Fallunterscheidungen mit **if..else** (wenn .. sonst) sind in jeder Programmiersprache bekannt. Die Syntax in PHP ist dieselbe wie z. B. in Javascript:

```
if (Bedingung) {
    Anweisungen des Wahr-Teils (der Bedingung)
}
else {
    Anweisungen des falsch-Teils
}
```

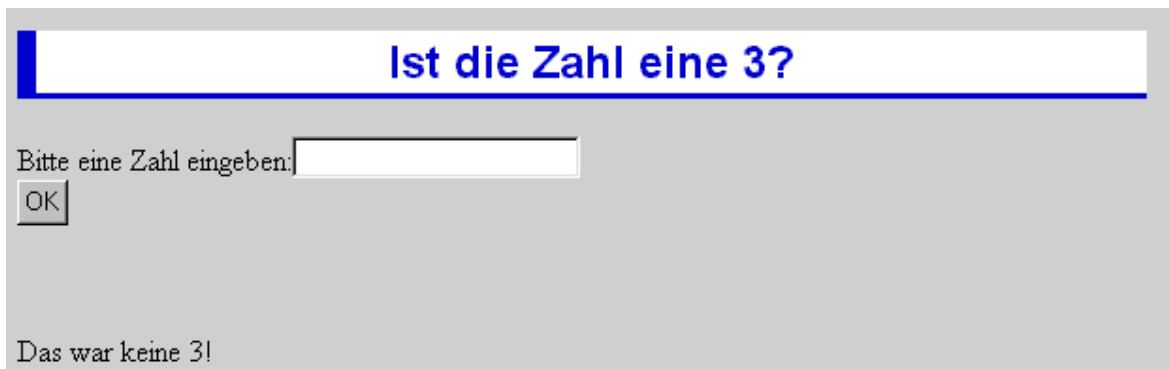
Wenn in der Verzweigung noch eine weitere Wenn-Prüfung erfolgt, dann ist dies z. B. so zu schreiben:

```
if (Bedingung) {
    Anweisungen des wahr-Teils (der Bedingung)
}
elseif {
    Anweisungen der 2. wenn-Bedingung)
}
elseif {
    Anweisungen der 3. wenn-Bedingung)
}
else {
    Wenn alle vorher genannten Möglichkeiten nicht zutreffen
}
```

Aufgabe: Erstellen Sie die Datei **15.php**:

```
<html>
<!-- 15.php -->
<link rel="stylesheet" href="standard.css" type="text/css">
<head>
<title>IF-Verzweigung</title>
</head>
<body>
<h1>Ist die Zahl eine 3?</h1>
<form action = "15.php" method = "post">
Bitte eine Zahl eingeben:<input type = "text" name="zahl"></input><br>
<input type="submit" name="OK" value="OK"></input>
</form>
<br><br>
<?php
if($zahl==3)
    {
        echo "Du hast eine 3 eingegeben!";
    }
else
    {
        echo "Das war keine 3!";
    }
?>
<a href="http://localhost/php.php">zurück zur Index-Seite</a>
</body></html>
```

Im Formular wird eine Zahl abgefragt. Dann prüft PHP, ob die Zahl eine 3 war oder nicht.



Ist die Zahl eine 3?

Bitte eine Zahl eingeben:

OK

Das war keine 3!

Erst, wenn eine 3 eingegeben wurde und auf OK geklickt wurde, wird die Ausgabe sein:

Du hast eine 3 eingegeben!

7.1 Vergleichsoperatoren und logische Operatoren

Die Bedingung beim "if" ist meist ein Vergleich zweier Werte oder Variablen.

Folgende **Vergleichsoperatoren** stehen zur Verfügung:

==	gleich
<	kleiner als
<=	kleiner oder gleich
>	größer
>=	größer oder gleich
!=	ungleich

Außerdem gibt es die folgenden **logischen Operatoren**:

&&	logisches UND
	logisches ODER (das Zeichen wird mit Alt GR und der Taste, auf der sich das < und > - Zeichen befindet, erzeugt)
!	logisches NICHT

Achtung: Eine der häufigsten Fehlerquellen ist das Verwecheln von Vergleich (==) und Zuweisung (=).

Aufgabe: Öffnen Sie die Datei mit dem Namen **14.php** und speichern Sie sie mit dem neuen Namen **16.php**

Es soll erreicht werden, dass erst dann, wenn im Formular ein Vorname eingetragen wurde, unten auch die Ausgabe "Hallo," sichtbar wird.



Wie heißt du?:

OK

Dies gelingt durch eine if-Verzweigung:

```
<?php
if($vorname=="") {
    echo "";
}
else
{
    echo "Hallo, <b>$vorname</b>,!";
}
?>
```

Aufgabe: Verändern Sie nach diesem Muster auch die Datei **15.php**. Bisher zeigt sie immer schon zu Anfang, wenn noch gar nichts eingegeben wurde, die Aussage:
Das war keine 3!

Speichern Sie das Ergebnis mit dem Namen **17.php**.