PHP

PHP (Abkürzung für „PHP: Hypertext Preprocessor“, ursprünglich „Personal Home Page Tools“) ist eine Skriptsprache mit einer an C bzw. C++ angelehnten Syntax, die hauptsächlich zur Erstellung von dynamischen Webseiten oder Webanwendungen verwendet wird.

PHP ist eine Skriptsprache, deren Befehle direkt in den HTML-Quelltext geschrieben werden können. Beim Aufrufen eines PHP-Skripts im Browser erfolgt die Ausführung des Skripts auf dem Webserver. Das Ergebnis wird im HTML-Format an den Browser zurückgegeben.

Um PHP-Skripte zu erstellen, braucht man zwei Dinge:

1. Die Webserver-Software – z.B. Apache und
2. PHP selbst

Zusätzlich eine MySQL Datenbank.

Sowohl dynamische Webseiten als auch Webanwendungen verwenden PHP zur Verarbeitung von Formulardaten, dem Generieren von HTML-Inhalten sowie zur Arbeit mit Datenbanken. Das trifft auch auf die meisten populären Content Management Systeme, wie WordPress, Joomla oder Drupal zu. Nicht zuletzt basieren darüber hinaus viele E-Commerce-Plattformen und Online-Shops auf PHP-Code.

Vorteile von PHP

- Einfache Syntax, die keine fortgeschrittenen Programmierkenntnisse erfordert
- Umfangreiche offizielle Dokumentation mit zahlreichen Ressourcen, die die Entwicklung mit PHP erleichtern
- Breite Unterstützung von den meisten Webhosting-Anbietern und oft serverseitig vorinstalliert
- Ständige Weiterentwicklung, zum Beispiel die Ergänzung um objektorientierte Programmierung
- Frei verfügbar und kostenlos nutzbar, da Open Source

Nachteile von PHP

- Geringe Skalierbarkeit, die bei großen, komplexen Anwendungen schnell an ihre Grenzen stößt
- Die weite Verbreitung macht PHP-Sicherheitslücken zum lohnenden Ziel
- Beschränkte Verwendung ausschließlich für Webseiten und Web-Anwendungen

Unterschied Programmiersprache und Skriptsprache

Programmiersprache:
Dabei wird der von Menschen lesbare Programmiercode durch ein spezielles Hilfsprogramm, einem Compiler, in Maschinencode umgesetzt. Dieser Vorgang heißt Kompilierung.

Skriptsprache:
Bei einer Skriptsprache werden dagegen die Anweisungen beim Aufruf zuerst von einem Interpreter sequenziell abgearbeitet und umgesetzt, und zwar bei jedem Aufruf erneut.

Bei den Skriptsprachen kann man zwei verschiedene Arten unterscheiden:

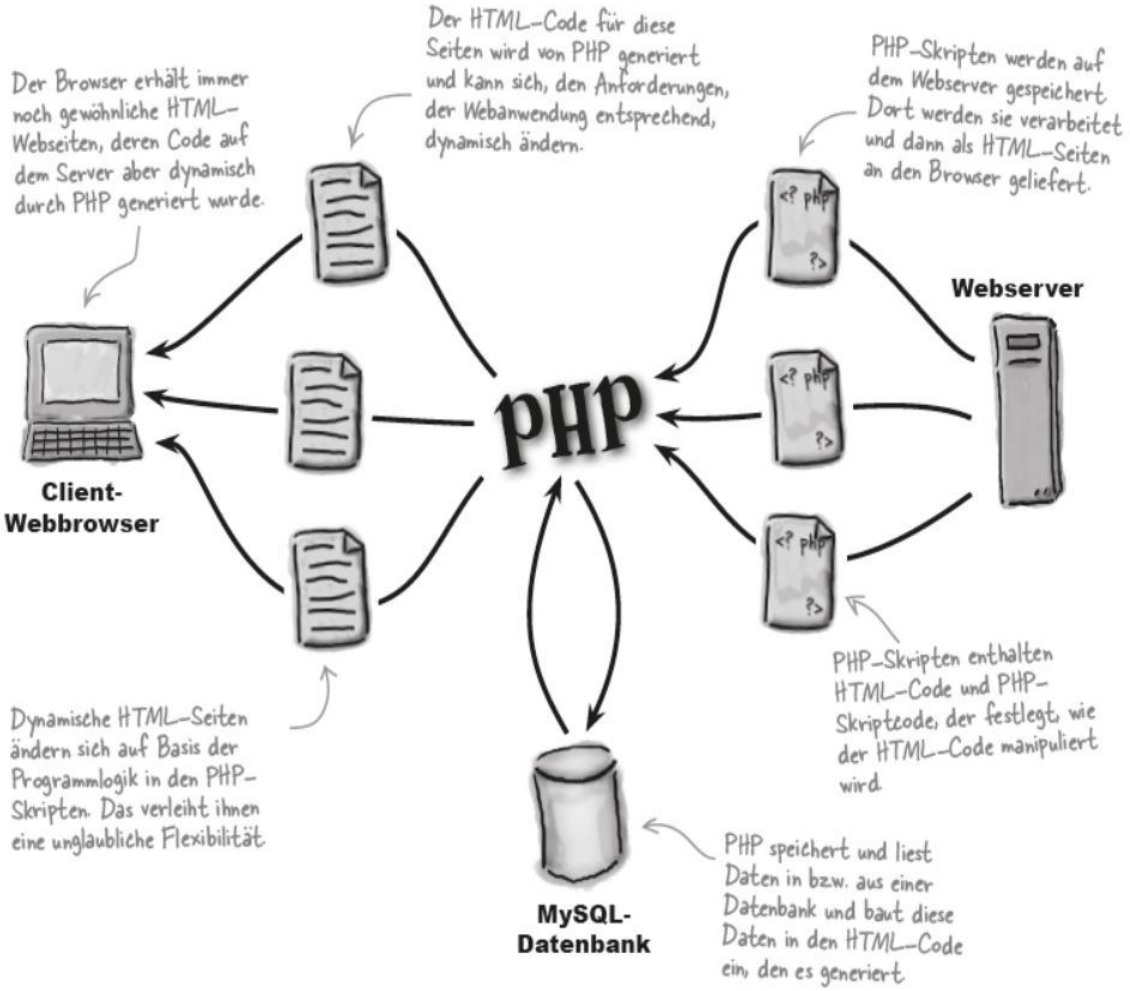
1. **Serverseitig**: dabei wird ein Script auf einem Server ausgeführt und an den Client nur als Ergebnis übermittelt. Genau das ist bei PHP der Fall. **PHP ist also eine serverseitige Skriptsprache.**

2. **Clientseitig**: dabei wird der Skriptcode direkt an den Client geschickt und erst dort ausgeführt. Das populärste Beispiel dafür ist JavaScript. JavaScript wird z.B. eingesetzt, um Pop-up-Fenster zu öffnen, um Formulareingaben zu prüfen oder für Verbesserungen an der Benutzeroberfläche, wie Tabs und Accordions. Es wird im Browser ausgeführt und kann vom Benutzer deaktiviert werden.

PHP macht Webseiten lebendig.

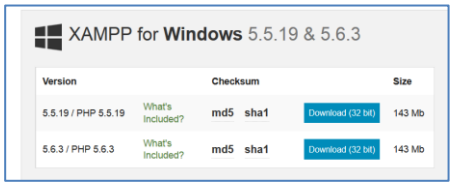
HTML ist statisch. Ein Browser fordert eine Seite an, der Server antwortet mit HTML. Ende der Geschichte. Sollen Websites zu interaktiven Webanwendungen werden, muss der Webserver eine neue, dynamischere Rolle übernehmen ... eine Rolle, die PHP möglich macht.

Mit PHP kann man den Inhalt von Webseiten auf dem Server manipulieren, bevor die Seite an den Clientbrowser geliefert wird.
Mithilfe von PHP kann der Webserver HTML-Seiten dynamisch generieren.
Das funktioniert so: Auf dem Server läuft ein PHP-Skript, das HTML-Code beliebig ändern oder erzeugen kann. Dem Browser wird weiterhin eine HTML-Webseite geliefert, aber er erfährt nichts davon, dass das HTML auf dem Server mit PHP bearbeitet wurde.

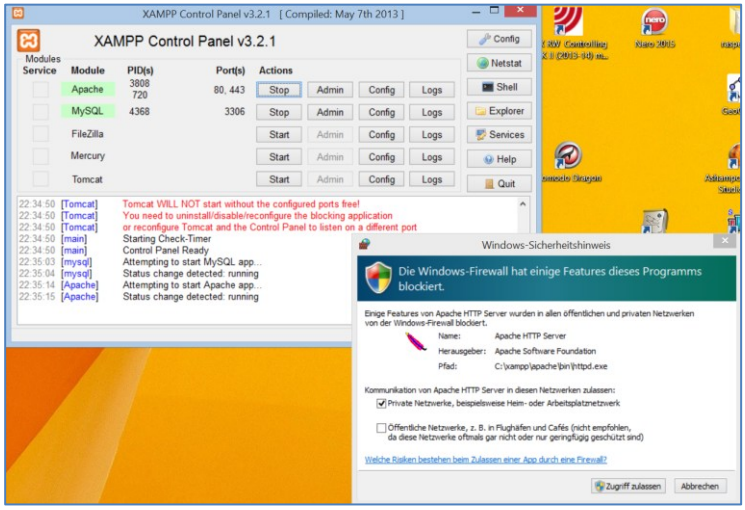

Der HTML-Code für diese Seiten wird von PHP generiert und kann sich, den Anforderungen, der Webanwendung entsprechend, dynamisch ändern.
PHP-Skripten werden auf dem Webserver gespeichert. Dort werden sie verarbeitet und dann als HTML-Seiten an den Browser geliefert.
Der Browser erhält immer noch gewöhnliche HTML-Webseiten, deren Code auf dem Server aber dynamisch durch PHP generiert wurde.
Webserver
PHP
Client-Webbrowser
Dynamische HTML-Seiten ändern sich auf Basis der Programmlogik in den PHP-Skripten. Das verleiht ihnen eine unglaubliche Flexibilität.
PHP-Skripten enthalten HTML-Code und PHP-Skriptcode, der festlegt, wie der HTML-Code manipuliert wird.
PHP speichert und liest Daten in bzw. aus einer Datenbank und baut diese Daten in den HTML-Code ein, den es generiert.
MySQL-Datenbank

PHP-Code braucht einen Server

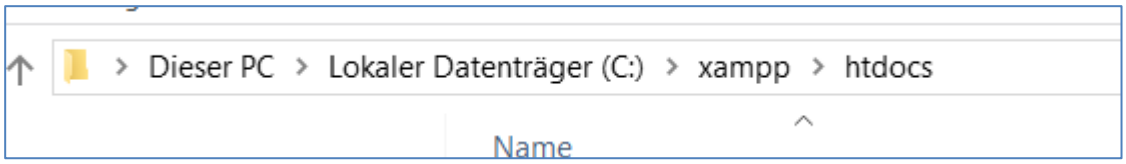
Am besten verwendet man XAMPP. Download von www.apachefriends.org
Zum Installieren nimm besser nicht das Verzeichnis c:\programm files, wenn UAC (User Account Control) aktiviert ist.
Mercury ist der Mailserver, den man hier nicht benötigt.



Eventuell muss man nach dem Klick auf „Start“ den „Zugriff zulassen“:



Damit XAMPP mit seinem Server die Dateien behandeln kann, müssen diese in einem bestimmten Ordner von XAMPP liegen.



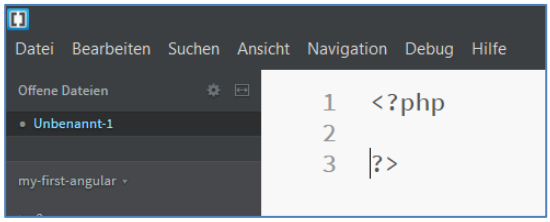
Erstelle daher den ersten Übungsordner im Ordner „htdocs“ mit dem Namen „php“.

Erstellen einer neuen Site:

PHP-Skripte lassen sich mit einem normalen Editor erfassen. Um bei längeren Skripten den Überblick zu behalten, empfiehlt sich der Einsatz eines Editors, der die einzelnen Sprachelemente farblich hervorhebt und die Zeilennummer anzeigt, z.B. Brackets.io, VisualStudio-Code, NetBeans oder www.eclipse.org .

Erstellen einer PHP-Seite mit der Endung .php

Mit dem Editor „visualStudio-Code“:
Datei / Neu
und gib den ersten PHP-Code ein:

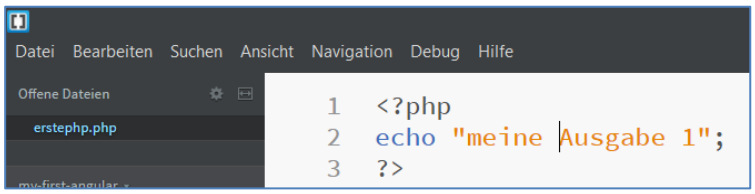


Name: „erstephp.php“ im Ordner „php“ von XAMPP / htdocs.

Der lokale Stammordner einer Site muss sich immer im Ordner C:\...\xampp\htdocs\ befinden.

Schreibe den ersten Text, der auch ausgegeben werden soll:

echo “meine Ausgabe 1“;

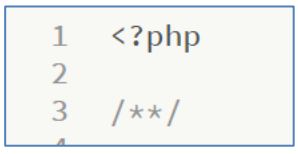


Kommentare:

Für Anmerkungen innerhalb des PHP-Scripts können Kommentare eingefügt werden. Steht der Kommentar in nur einer Zeile, wird er mit zwei Schrägstrichen eröffnet, ist er allerdings mehrere Zeilen lang, muss er mit /* bzw. */ eingegrenzt werden.
Kommentare können auch nach abgeschlossenen Anweisungen stehen.

Tipp: automatischer Kommentar einfügen lassen – drücke die Tasten

- STRG + SHIFT + Raute

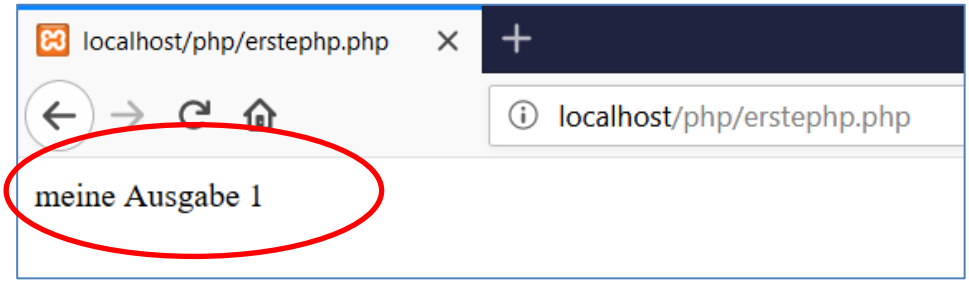


z.B.:

```
<?php
$a = 7;     // einzeiliger Kommentar
 /* mehrzeiliger
Kommentar */
?>
```

Anzeige im Browser – wenn der Sever in XAMPP läuft:

Für die Anzeige im Browser ist die Adresse einzugeben die mit http://localhost beginnt, gefolgt vom entsprechenden Verzeichnis- und Dateinamen, wie im obigen Beispiel:
http://localhost/php/erstephp.php



PHP-Code:

- Jedes PHP-Script muss innerhalb des PHP-Tags stehen: Standardmäßig wird PHP-Code mit **<?php** eingeleitet und mit **?>** beendet. Alles, was zwischen diesen beiden Markierungen steht, wird vom Interpreter ausgewertet.

- **Alle Zeilen** (außer if/else und Schleifen) werden mit einem **Strichpunkt abgeschlossen**.

PHP-Quellcode besteht aus einer Folge von Anweisungen, die nacheinander abgearbeitet werden. Einzelne Anweisungen müssen voneinander getrennt werden. In PHP wird dazu das Semikolon (**Strichpunkt**) verwendet.

- Die Namen von PHP-Variablen müssen mit einem Dollarzeichen $ beginnen.
- Wenn eine Website PHP-Code enthält, muss sie die Dateinamenserweiterung .php tragen, nicht .html.

Erstes php-Skript in HTML einbinden:

Damit das Beispiel funktioniert, ist zweierlei nötig:

1. Die Datei muss im richtigen Verzeichnis abgespeichert werden (xampp/htdocs/php) und
2. Es muss die Endung .php haben

Ausgaben erfolgen mittels des Befehls **echo**, wobei die auszugebenden Zeichen von Anführungszeichen eingeschlossen sein müssen. Auch Html-Tags, wie z.B.
, können in den echo-Befehl eingebunden werden.

```
1  <?php
2  echo "meine Ausgabe 1";
3  ?>
```

Man kann z.B. auch für kursiv einfügen:

echo “Ausgabe 01
“;

Die PHP-Anweisung „echo“ gibt den angegebenen Text auf dem Bildschirm aus. Der Text muss in Anführungszeichen geschrieben werden. Statt „echo“ kann man auch „print“ verwenden. Falls der Text HTML-Markierungen beinhaltet, z.B.
, werden diese ausgeführt.

Kontrolle im Browser (Seitenquelltext):

Wechsle in den Quellcode, in Firefox etwa über Extras/Entwickler-Werkzeuge/Seitenquelltext. Hier sieht man **KEINEN PHP-Code, sondern nur HTML-Code**. Wenn das so ist, hat alles geklappt. Im Browser sieht man nämlich das, was der PHP-Interpreter auf dem Server erzeugt hat, einen reinen HTML-Code ohne PHP-Befehle:

- **Leerzeichen** und neue Zeilen sind für PHP nicht relevant. Sie sind aber natürlich wichtig für die Lesbarkeit des Skripts. Man hat sich auch geeinigt, den Code in Kleinbuchstaben zu schreiben.

- Die PHP-Befehle **kann man an beliebigen Stellen im HTML einfügen** – immer da, wo man sie brauchen. Also dort, wo man beispielsweise einen Wert aus der Datenbank ausgeben oder das Ergebnis einer Berechnung anzeigen lassen will.

- **Ausgabe mit „echo“ oder mit „print“**
  Manche Programmierer bevorzugen die Funktion „print“, die fast genauso arbeitet wie echo. Echo ist aber in der Praxis ein bisschen flexibler, deshalb wird es öfters verwendet.

Variablen:

Eine Variable dient als Speicherort für einen Wert, also Platzhalter für unterschiedliche Daten. Einer Variablen kann jederzeit ein neuer Wert zugewiesen werden, sie ist also, wie der Name schon sagt, variabel.
Variablen sind beispielsweise nötig, um Eingaben der Benutzer weiterzuverarbeiten. Man weiß ja noch nicht, was die Benutzer eingeben, möchte aber trotzdem darauf zugreifen, um die Inhalte auszugeben.

In PHP sind Variablennamen am **einleitenden $-Zeichen** zu erkennen. Variablen können beispielsweise Text, Ganzzahlen oder Gleitkommazahlen enthalten. PHP weist einer Variablen **automatisch** den gerade benötigten Datentyp zu.
Datentypen können z.B. sein Ganzzahl (integer), string (z.B. Training), Arrays, Objekte.
Man muss die Datentypen nicht angeben (wie z.B. in Java oder C#). Intern werden diese schon unterschieden.

Einer Variable wird mit dem „**Istgleichzeichen**“ = ein Wert zugewiesen, wobei auf der linken Seite die Variable und rechts der gewünschte Wert stehen muss. Es bedeutet „enthält den Wert“.
Die Leerzeichen vor und nach dem Istgleichzeichen haben lediglich optische Funktion.

Beispiel:
```
$name = "Max Mustermann";
$alter = 30;
```

Regeln:

- Die **Groß- und Kleinschreibung ist relevant**. So sind $meineVariable und $MeineVariable unterschiedliche Variablen.
- Nach dem $ darf nicht direkt eine Zahl folgen
- Leerzeichen, Punkte, Ausrufezeichen oder Bindestriche sind im Variablennamen nicht erlaubt. Statt des Leerzeichens nimmt man am besten einen Unterstrich, z.B. $brutto_preis.

Zeichenketten (Strings) müssen in

- doppelten Hochkommata (“ “) oder in
- einfachen Hochkommata (‘ ‘) eingeschlossen werden.

Inhalt von Variablen ausgeben:
Den Inhalt von Variablen per echo ausgeben

- echo $name;

Um ganze Sätze zu schreiben kann man Textinhalt mit dem Inhalt von Variablen kombinieren. Die richtige Schreibweise ist dabei das doppelte Hochkomma. Nur damit wird die „Variablen-Interpolation“ richtig durchgeführt:

Nur unter doppelten Hochkomma kann man Variablen problemlos einbauen, wie z.B. hier:

```php
$name = "Max Mustermann";
$alter = 30;

echo "<p>Mein Name ist $name und ich bin $alter Jahre alt.</p>";
```

Tipp:

Man kann auch die **verkürzte Schreibweise für das „echo“** verwenden.

<?= $name?>

Direkt nach dem <? ein = Zeichen und dann das, was ausgegeben werden soll.

Keine Regel aber sinnvolle Konvention ist:

- Nutze in Variablennamen nur Kleinbuchstaben
- Trenne die Wörter in aus mehreren Wörtern bestehenden Variablennamen mit Unterstrichen

Übung:

Schreibe darunter: beachte die einfachen Wechsel zwischen HTML und PHP

```php
<html>
<head>
<title>erste PHP</title>
</head>
<body>
    <h1>hi, das ist PHP</h1>

    <?php echo "hallo";
    $schueler1 = "Katja";
    $schueler2 = 'Kevin';
    ?>

    <br><br>

    <?php
    echo "$schueler1 sitzt vor $schueler2";
    ?>
```

Damit ein Zeilenumbruch die Ausgabe verschönert verwende ein
:

Ergebnis:

Operatoren

Operatoren benötigt man für Berechnungen und zur Verkettung von Zeichenketten.

Mit Operatoren Werte zuweisen und vergleichen

- Gleichheitszeichen und
- doppeltes Gleichheitszeichen:

Bei dem Ausdruck

```
$name = "Max Mustermann";
```

wird der linken Variablen der rechtsstehende Wert zugordnet. Das **einfache Gleichheitszeichen** wird hier als Zuweisungsoperator bezeichnet.

Im Unterschied zum einfachen Gleichheitszeichen dient das **doppelte Gleichheitszeichen als *VERGLEICHSOPERATOR*:**

Mit $a == $b wird überprüft, ob die Werte der beiden Variablen gleich sind. Ist dies der Fall, ergibt der Ausdruck den ***Wert TRUE***. Ansonsten wird FALSE ausgegeben.

Variablen ausgeben:

Sobald eine Variable eingeführt wurde, kann sie im Script verwendet und auch in einem String mit „echo“ ausgegeben werden: Dabei gibt es zwei Möglichkeiten und eine 3, etwas umständlichere, namens „verketten“

1. Man kann die Variable direkt mit echo ausgeben, dann, wenn kein String nötig ist – mit oder ohne Anführungszeichen:

   - echo $Groesse;
     oder
   - echo “$Groesse“;

2. Man kann sie als **Teil eines Textes** innerhalb von (doppelten) Anführungszeichen setzten. In diesem Fall spricht man davon, dass die Variable „interpoliert“ wird. Das funktioniert nur bei doppelten Anführungszeichen. Diese Vorgehensweise nennt man die soge-

nannte **„Variable Interpolation“: Das spart viel Schreibarbeit und ist eine Spezialität von PHP.**

Man setzt die Variablen einfach in die Zeichenkette hinein:

```
echo "Die Größe beträgt $Groesse cm. ";
```

3. Die kompliziertere Möglichkeit eine Variable auszugeben, nennt man das sogenannte **Verketten.** Unter Verketten versteht man das Verbinden von zwei oder mehr Zeichenketten. Das wird normalerweise mit einem sogenannten Verkettungsoperator, **einem PUNKT** erledigt. Die folgende Variante ist aus anderen Programmiersprachen bekannt und verwendet immer den Punkt zur Verkettung. Dabei wird die Variable zu ihrer linken und rechten Seite je mit dem umgebenden String verkettet.

```
echo "Die Größe beträgt " . $Groesse . " cm. <br>";
```

Wie bereits oben geschrieben, ist diese komplizierte Schreibweise in PHP meistens **nicht nötig**. Die Ausgabe, das Ergebnis beider Varianten (Interpolation und Verketten) ist gleich. Das funktioniert nur mit doppelten Anführungszeichen!

Aufpassen bezüglich Leertaste und Abstände – sonst gibt’s Fehler oft bei den Beistrichen.

Achtung: Innerhalb der doppelten Anführungszeichen dürfen dazwischen keine weiteren doppelten Anführungszeichen vorkommen!

- Man kann dazwischen einfache Anführungszeichen verwenden
- oder eine Maskierung verwenden z.B. /“

Verkettung mit einer Funktion inkl. Funktion date()

Eine Funktion muss man mit dem PUNKT verketten.
Zum Ausgeben der aktuellen Jahreszahl dient die Funktion date("y").
Achte beim Verketten, dass die Leerzeichen mitberücksichtigt werden.
Man muss die VERKETTUNG benutzen.

Das aktuelle Datum ausgeben:

echo date("j.n.Y");
Man kann hier beide Anführungszeichen benutzen, die einfachen oder die doppelten.

```
19    echo "<br>";
20    echo date("j.n.Y");
```

j.....Monatstag ohne führende Null (1-31)
d....Monatstag mit führender Null (01-31)
n....Monatszahl ohne führende Null (1-12)
D...Tag ausgeschrieben
m...Monat mit führender Null
M...Monat ausgeschrieben
y....Jahreszahl mit 2 Stellen (19)
Y....Jahreszahl mit 4 Stellen (2019)

Ergebnis:

Hallo Welt!

Mein Name ist Max Mustermann und ich bin 30 Jahre alt.

17.11.2025

Übung: Gib folgendes aus mit 2 x
 dazwischen

Heute ist das Datum 17.11.2025

Andere Schreibweise ist folgende Mon Nov 2025

Lösung:

```
21    echo "Heute ist das Datum " . date("j.n.Y");
22    echo "<br><br>";
23    echo "Andere Schreibweise ist folgende " . date('D M Y');
24
```

Sonderzeichen in Anführungszeichen – Das Escape-Zeichen das ist der „Backslash“

Möchte man innerhalb von doppelten Anführungszeichen ein Zeichen ausgeben, das eigentlich für die Syntax reserviert ist, z.B. Anführungszeichen dann muss man das Maskieren, indem man einen Backslash davorstellt.

```
echo "Mein Name ist \"Hase\" und habe 5,00 $."
```

Das benötigt man z.B. auch, wenn man Attributwerte von HTML verwendet, das selbst in Anführungszeichen geschrieben wird:

Hier soll ein Bild ausgegeben werden, daher muss man die doppelten Anführungszeichen dazwischen schützen (=maskieren):

```
echo “<img src=\“wiese.jpg\“ widht=\“200\“ height=\“auto\“>“;
```

Man kann anstatt die doppelten Anführungszeichen über \ zu maskieren, sie auch mit einfachen Gedankenstrichen maskieren:

```
echo “<img src=‘wiese.jpg‘>“;
```

Die Escape-Sequenz für einen Zeilenumbruch ist \n.

An jeder Stelle, an der in einem Text \n erscheint, wird ein Zeilenumbruch eingefügt, der bewirkt, dass mit der Ausgabe des nachfolgenden Inhalts am Anfang der nächsten Zeile fortgefahren wird.

PHP kennt nur einen sehr beschränkten Satz von Escape-Sequenzen. Wie bei \n werden sie mit einem Backslash eingeleitet, der das nachfolgende Zeichen maskiert oder schützt, d.h. ihm seine gewöhnliche Bedeutung für den PHP-Interpreter nimmt, wenn dies zu Problemen führen könnte. Beispiele sind die Escape-Sequenzen für einfache Anführungszeichen (\'), doppelte Anführungs-zeichen (\") und den Zeilenumbruch (\n).

Funktion: Mail()

Um mit PHP ein Mail zu senden, verwende die **vorgefertigte Funktion „mail()“**

Folgende drei Informationen werden von der Funktion mail() verlangt und **müssen angegeben** werden (verpflichtend):

- $an die Empfängeradresse
- $betreff der Betreff der Nachricht
- $msg der Inhalt der Nachricht (message)

Man kann noch weitere Argumente verwenden:

- From: $email Bei der Angabe der Senderadresse muss der E-Mail-Adresse der Text „From:“ vorangestellt werden.
  Die Senderadresse kann aus der Formulareingabe des Kunden genommen werden, wenn dieser im Formular diese eingeben musste:
  $email = $_POST[“email“];

Beispiel:

```
$an = "josefeberhart@gmx.at";
$betreff = "erstes PHP-Mail";
$msg = "Sehr geehrter Kunde. Danke  $name  für dein Mail. \n"  .
        "Mit freundlichen Grüßen.";
$from = „From: Kontaktformular. Folgender Interessent: $email“;
mail($an, $betreff, $msg, $from);
```

```
 9  $msg = "Mein Name ist $vor $nach und ich benötige Infos. Meine E-Mail
    lautet $email";
10
11  mail($an, $betreff, $msg);
12
13  echo "<h2>Ihre Mail wurde an $an weitergeleitet mit folgendem Inhalt: <br>
    Mein Name ist $vor $nach und ich benötige Infos. <br> Meine E-Mail lautet
    $email</h2>";
```

Ergebnis:
Mail-Acccount:

Eine bessere Möglichkeit E-Mails zu senden bietet ein sogenannter „PHPMailer“.

https://github.com/phpmailer/phpmailer

Das wird hier aber nicht gezeigt.

Rechnen mit PHP

Die Grundrechnungsarten gelten auch in PHP. PHP arbeitet mit den vertrauten Rechenregeln, es werden also geklammerte Ausdrücke zuerst berechnet. Außerdem gilt die „Punkt vor Strich"-Regel, es werden also zuerst Multiplikation bzw. Division und anschließend Addition und Subtraktion ausgeführt.
Beachte: Das Komma in einer Zahl wird – wie bei allen Programmier- und Scriptsprachen – mit einem Punkt notiert.

Beispiel:
Arithmetische Operatoren: Er arbeitet 20 Stunden pro Woche und verdient pro Stunde Euro 20,-. Wie viel verdient er im Jahr, wenn er wirklich jede Woche (52) arbeitet.

```
<?php
$stunden = 20;
$wochen = 52;
$betrag = 20;
$gesamt = ($stunden * $wochen * $betrag);
echo "Er verdient $gesamt Euro.";
?>
```

Speichern und im Browser ansehen.
Ergebnis:

Beispiel:
Arbeite weiter in der „rechnen.php". Estelle folgende einfache Rechenoperationen:

Beachte den Code im <echo>
Die Variablenwerte werden innerhalb der Anführungszeichen ausgegeben, die Berechnungen mit **einem Komma hinzugefügt** und anschließend folgt, nach einem weiteren Komma und wieder in Anführungszeichen, der HTML-Code
 für einen Zeilenumbruch.

```
25  <h1> Einfache Rechenoperationen</h1>
26  <p>Gegeben sind die beiden Werte 56 und 18.</p>
27  <?php
28  $a = 56;
29  $b = 18;
30  echo "$a + $b = ", $a+$b, "<br>";
31  echo "$a - $b = ", $a-$b, "<br>";
32  echo "$a / $b = ", $a/$b;
```

Weiteres Beispiel:

```php
<?php
$faktor = 20;
$euro = 10;
$erg = $faktor*$euro;
echo "$erg";

?>
```

Vor der Berechnung wurde der <php>-Code beendet, damit der HTML-Code die Überschrift ausgeben kann.

Formatierung der Zahlen:

Funktion number_format()

```php
$a = 56;
$a = number_format($a,2,",",".");
```

Diese Funktion wird über bis zu vier Parameter gesteuert, da man in der Klammer übergibt, jeweils mit einem Beistrich getrennt:

- **Wert**: An erster Stelle steht der Wert, den man formatieren möchte
- **Nachkommastellen**: nach einem Komma folgt die Angabe der Nachkommastellen, die man haben möchte. Meistens wird dies „2“ sein. Dabei rundet die Funktion automatisch auf bzw. ab.
- **Dezimaltrennzeichen**: an dritter Position gibt man in Anführungszeichen an, welches Trennzeichen für die Dezimalstelle man haben möchte. Standardmäßig ist dies der Punkt, es wird aber bei uns ein Komma benötigt: ","
- **Tausendertrennzeichen**: Hier wird in Europa der Punkt verwendet, im angelsächsischen Raum ein Beistrich: "."

Übung:

```php
28      <h2>Formatierung mit number_format</h2>
29      <?php
30      $divident = 56;
31      $divisor = 18;
32      $quotient = $divident/$divisor;
33
34      echo "$quotient";
35      ?>
```

Ergebnis:

Formatierung mit number_format

3.1111111111111

Lösung:
Daher muss man für die Division eine zusätzliche Variable erstellen und diese formatieren: füge daher wie in Zeile 33 ein:

```
30      $divident = 56;
31      $divisor = 18;
32      $quotient = $divident/$divisor;
33      $quotient = number_format($quotient,2,",",".");
34
35      echo "$quotient";
```

Übung „zweitephp.php“:

Schreibe in der PHP-Datei weiter. Damit man schöne <h1>-Überschriften nutzen kann, unterbrich den PHP-Code regelmäßig in diesem Beispiel:

```
7   <br><br>
8   <h1>Berechnung</h1>
9   <?php
10  $zahl = 57000000000 / 8.5;
11  $zahl = number_format($zahl,2,",",".");
12  echo $zahl;
13  ?>
14  <br><br>
```

Ergebnis:

Berechnung

6.705.882.352,94

Die Zeichen bei Milliarde und Million werden automatisch auch richtig als Punkt dargestellt.

Übung:
Berechne folgenden Sachverhalt: Franz hat 200 Euro auf dem Sparbuch und erhält dafür 3% Zinsen (=0,03). Wie viel Kapital hat er nach einem Jahr inklusiv der Zinsen auf dem Sparbuch?

HÜ:
Berechnung des Mehrwertsteueranteils
Berechne bei einem Steuersatz von 20% bei einem gegebenen Bruttobetrag von € 196,- sowohl den Netto- als auch den Steuerbetrag.
Verwende dabei auch die Funktion number_format().

Weitere Operatoren:

Beispiel	Beschreibung
`$a = $b; $a`	wird der Wert von $b zugewiesen
`$a - $b;`	Differenz von $a und $b
`$a + $b;`	Summe von $a und $b
`$a * $b;`	Produkt von $a und $b
`$c = $a . $b . " ";`	Verknüpfung von Zeichenketten
`$a .= $b;`	An $a wird der Wert von $b angehängt
`$a == $b`	TRUE, wenn Wert von $a gleich $b ist
`$a != $b`	TRUE, wenn Wert von $a ungleich $b ist
`$a > $b`	TRUE, wenn Wert von $a größer $b ist
`$a < $b`	TRUE, wenn Wert von $a kleiner $b ist
`$a <= $b`	TRUE, wenn Wert von $a kleiner oder gleich $b ist
`$a >= $b`	TRUE, wenn Wert von $a größer oder gleich $b ist
`$a++`	Wert von $a wird um 1 erhöht
`$a--`	Wert von $a wird um 1 verringert
`$a and $b`	TRUE, wenn sowohl $a als auch $b jeweils TRUE sind
`$a && $b`	(alternative Schreibweise)
`$a or $b`	TRUE, wenn entweder $a oder $bTRUE sind
`$a || $b`	(alternative Schreibweise)

Operator Ist-Gleich:

Der Operator Ist-Gleich wird mit zwei Istgleichzeichen dargestellt, um es von einer Zuweisung (Variablen) zu unterscheiden:

==

Operator Ungleich (Negation):

Zur Negation eines Ausdrucks dient das Ausrufezeichen. Am häufigsten wird es dazu benutzt, die Ungleichheit beziehungsweise Nicht-Identität von Variablen festzustellen.

Der Operator Ungleich kann mit zwei völlig gleichrangigen Operatoren verwendet werden, nämlich:

!= oder <>

Beispiel: $x != $y bedeutet: Der Wert der Variablen $x ist ungleich dem Wert der Variablen $y.

Punkt vor Strich

Wenn man Berechnungen im PHP-Code durchführen, dann gilt, so wie man es erwarten würde, die Regel »Punkt vor Strich«. Das heißt, dass in einem Ausdruck wie

$i = 5 – 3 * 2;

zuerst die Multiplikation ausgeführt wird (3 * 2) und danach die Subtraktion. Deswegen erhält im obigen Beispiel $i den Wert -1. Wenn man hingegen will, dass zuerst eine andere Operation durchgeführt werden soll, muss man Klammern einsetzen:

```
$k = (5 - 3) * 2;
```

Jetzt wird zuerst 5 – 3 berechnet und das Ergebnis mit 2 malgenommen, $k erhält also den Wert 4.

Operator Inkrement und Dekrement:

Dadurch wird eine Variable um eins erhöht bzw. verringert.
Operatoren: ++ oder --

Beispiel: Ergebnis ist 11.

```
<?php
$a = 10;
echo ++$a;
?>
```

Wichtig ist der Standort des ++. Steht der Operator ++ nach dem $a vom echo, dann wird zwar der Wert 10 ausgegeben, aber $a erhält den neuen Wert 11 zugeteilt, der für die nächste eventuelle Berechnung 11 ist.

Ausgabe beider Werte:

```
<?php
$a = 10;
echo $a++ . “<br>“;
echo $a;
?>
```

Ergebnis: