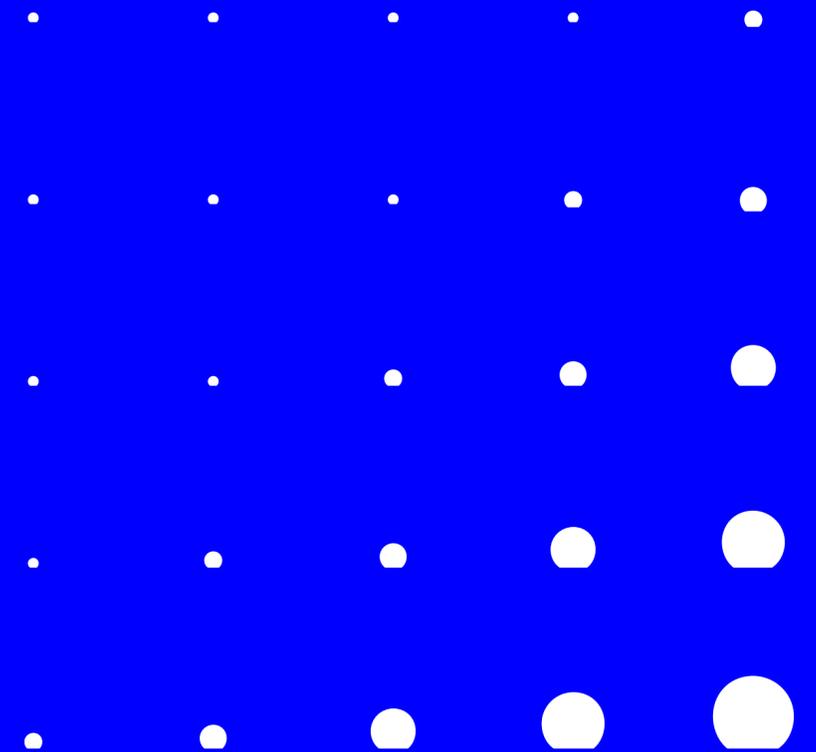


PHP

erste Schritte





Was ist PHP?

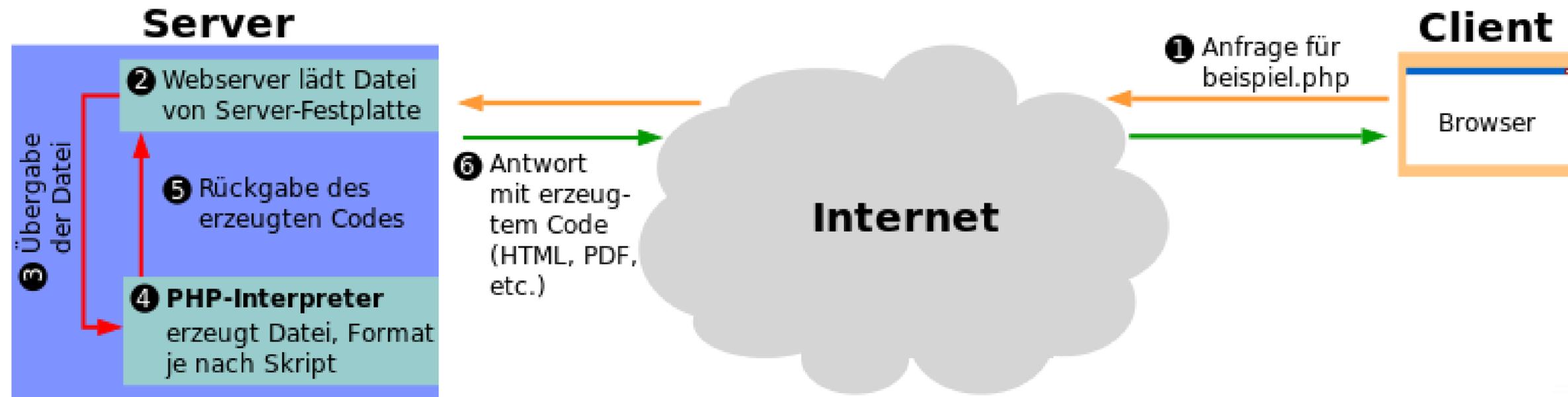
- PHP: Hypertext Preprocessor, ursprünglich Personal Home Page Tools ist eine server-seitige Skriptsprache mit einer an C und Perl angelehnten Syntax, die zur Erstellung dynamischer Webseiten oder Web-Anwendungen verwendet wird.
- PHP kommt auf über 70% (Stand Februar 2019, Quelle: W3Techs) aller Websites als serverseitige Programmiersprache zum Einsatz.
- Erreicht wird dies insbesondere durch beliebte Content-Management-Systeme wie WordPress, allein WP verwenden ca. 33% aller Webseiten.
- PHP ist dynamisch typisiert.
- PHP zeichnet sich durch eine
 - breite Datenbankunterstützung,
 - Internet-Protokolleinbindung sowie durch
 - die Verfügbarkeit zahlreicher Funktionsbibliotheken aus.



Was ist PHP?

- Eine PHP-Datei wird vom PHP-Interpreter ausgeführt!
 - Sie ist keine HTML-Datei für den Browser des Clients!
 - Sie wird vom Web-Server an den PHP-Interpreter übergeben!
 - Die Ausgabe des PHP-Interpreters ist normalerweise eine HTML-Seite.
- Der Text außerhalb von `<?php ... ?>` wird in die Ausgabe kopiert und damit zum Client gesendet.
- Auch Leerzeichen und Zeilenumbrüche werden in die Ausgabe kopiert!

Was ist PHP?



- Der Aufruf einer .php-Datei über den Client sorgt im Gegensatz zu einem .html-Aufruf nicht dafür, dass der Inhalt der Datei direkt an den Client in Form einer HTTP-Response zurückgegeben wird.
- Die .php-Datei wird statt dessen an einen PHP-Interpreter auf dem Server geleitet.
- Dessen Ausgabe wird dann erst in Form eines HTTP-Responses an den Client gesendet.

Was ist PHP?

- Wenn der Webserver richtig konfiguriert ist, wird PHP-Code also niemals als Text an den Client gesendet.
- PHP-Code wird dann ausschließlich serverseitig interpretiert und lediglich das Ergebnis davon an den Client gesendet.
- Ein einfaches Beispiel ist folgender Funktionsaufruf: `date("H:i:s")`
- Das Ergebnis dieses Funktionsaufrufs ist die aktuelle Serverzeit.
- Wird dieser Funktionsaufruf in eine .php-Datei eingebunden, wird lediglich das Ergebnis davon zum Client zurückgesendet: `10:57:37`
- Das erklärt auch, warum PHP-Code niemals als Text zum Client gesendet werden darf, denn:
- Womöglich werden Passwörter oder Zugangsdaten in .php-Dateien gehalten.

Konstanten

- Konstanten können über `define(...)` definiert werden
- einfache Ausgaben erfolgen in PHP mit `echo`

```
define("KONSTANTENNAME", "Ich bin der Inhalt");
```

```
echo KONSTANTENNAME;
```

- Ausgabe: `Ich bin der Inhalt`
- Konstanten sind global, d.h. überall auf dieser Seite sichtbar
- der Wert der Konstanten lässt sich nicht mehr ändern
- ein erneutes `define()` führt zu einer Fehlermeldung
- Konstanten können nicht mehr entfernt werden

Variablen & Datentypen

- es gibt keine unmittelbar sichtbaren Datentypen
- es ist keine Variablendeklaration nötig
- Variablennamen beginnen immer mit \$
- PHP typisiert die Variablen intern dynamisch
 - Boolean - bool mit den beiden Konstanten true und false
 - Ganzzahl - int, integer, long
 - Fließkommazahl - float, real, double
 - Zeichenkette – string
 - Arrays - array (indiziert oder assoziativ)
 - Objekt-Datentyp

Erster PHP-Code: Serverseitig

- PHP-Code beginnt immer mit `<?php` und endet mit `?>`
- PHP-Code kann direkt in das HTML-Dokument eingebunden werden.
 - Zu beachten ist allerdings, dass die Datei die Endung `.php` haben muss.
 - Hat die Datei stattdessen die Endung `.html`, wird der PHP-Code nicht interpretiert und direkt an den Client gesendet.

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Titel</title>
  </head>
  <body>
    <h1><?php echo "Ich bin eine Überschrift"; ?></h1>
  </body>
</html>
```

Ich bin eine Überschrift

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Titel</title>
  </head>
  <body>
    <h1>Ich bin eine Überschrift</h1>
  </body>
</html>
```

Alternative Erstellung des Beispiels: Serverseitig

- Die HTML-Elemente können auch direkt mit PHP ausgegeben werden.
- Dadurch wird der Code allerdings sehr unübersichtlich, weil HTML-Elemente lediglich als Zeichenketten betrachtet werden.
- Viele Entwicklungsumgebungen erkennen dann keine Fehler mehr im HTML-Code, da Zeichenketten nicht untersucht werden.

```
<?php
$ausgabe = "Ich bin eine Überschrift";
echo '<!DOCTYPE html>';
echo '<html lang="de">';
echo '<head>';
echo '<meta charset="utf-8" />';
echo '<meta name="viewport" content="width=device-width, initial-scale=1.0" />';
echo '<title>Titel</title>';
echo '</head>';
echo '<body>';
echo '<h1>' . $ausgabe . '</h1>';
echo '</body>';
echo '</html>';
?>
```

Konstanten, Variablen & Datentypen

- der Typ Boolean hat Besonderheit
 - 1 ist true (Zahl 1)
 - "" ist false (leerer String)
- Man kann den Datentyp einer Variablen prüfen mit...
 - is_bool(\$var) - Boolean
 - is_integer(\$var) - Ganzzahl
 - is_real(\$var) - Fließkomma
 - is_numeric(\$var) - Ganzzahl oder Fließkomma
 - is_string(\$var) - String
 - is_array(\$var) - Array
 - is_object(\$var) - Objekt
 - gettype(\$var) - Liefert den Typ als String

PHP und Debugging-Problematik

- Da false bei PHP lediglich eine leere Zeichenkette ist, haben PHP-Einsteiger oft Probleme beim Debuggen:

```
<?php
```

```
$ja = true;  
$nein = false;
```

```
echo $ja;  
echo "<br>";  
echo $nein;
```

```
?>
```

- Ausgabe: 1



Existenz von Variablen

- Der Zugriff auf nicht initialisierte Variablen ist verboten!
- Mit `isset` kann man Variable prüfen, es liefert
 - `true`, wenn diese Variable einen Wert hat.
 - `false`, falls die Variable nicht gesetzt wurde.
- Mit `unset` kann man eine Variable löschen.
 - PHP löscht Variablen automatisch am Ende der Sichtbarkeit.

```
<?php
```

```
echo isset($a); // -> false ()  
$a = "Hallo";  
echo isset($a); // -> true (1)  
unset($a);  
echo isset($a); // -> false ()
```

```
?>
```

Variablen auslesen mit var_dump

```
<?php
```

```
$a = 4;  
echo(var_dump($a));  
echo(' <br/>');
```

```
$a = 9.9;  
echo(var_dump($a));  
echo(' <br/>');
```

```
$a = false;  
echo(var_dump($a));  
echo(' <br/>');
```

```
?>
```

```
int(4)  
float(9.9)  
bool(false)
```

PHP und Debugging: Die Lösung

- `var_dump` gibt sowohl den Datentyp als auch den Wert der Variable an, dies löst die Problematik der fehlenden Ausgabe für `false`:

```
<?php
```

```
$ja = true;
```

```
$nein = false;
```

```
echo var_dump($ja);
```

```
echo "<br>";
```

```
echo var_dump($nein);
```

```
?>
```

- Ausgabe:

```
bool(true)  
bool(false)
```

Zeichenketten

- Begrenzer für Strings ist " oder '.
- "...": Variablen und Sonderzeichen darin werden ausgewertet.

```
$ausgabe = "Prof.";  
echo "$ausgabe Dr. Frank Dopatka";
```
- '...': Variablen und Sonderzeichen werden nicht ausgewertet.
- "\$ausgabe" würde genau so ausgegeben werden.
- Strings können Strings mit dem jeweils anderem Begrenzer enthalten.

```
echo '';  
echo "<img src='portrait.jpg'>";
```
- Ab PHP-Version 5.6 wurde der Standardzeichensatz von ISO-8859-1 auf UTF-8 geändert.

Zeichenketten

- Im Gegensatz zu vielen anderen Programmiersprachen werden Zeichenketten in PHP mit Punkten verkettet:

```
$ausgabe = "Prof.";  
echo $ausgabe . " " . "Frank Dopatka";
```

- Zeichenketten können in PHP als Array von Zeichen betrachtet werden:

```
$ausgabe = "Prof.";  
echo $ausgabe[0];
```

- Typische Fehler:
 - "Hello " + "World" ergibt 0
 - "Hello" . "World" ergibt "HelloWorld"
 - "Hello " . "World" ergibt "Hello World"

Funktionen auf Zeichenketten: Suchen und Vergleichen

Funktion	Bedeutung
\$wert=strlen(\$str)	gibt die Anzahl der Zeichen in \$str zurück
\$wert=strpos(\$str,\$such,\$offset)	gibt die 1. Position von \$such in der Zeichenkette \$str ab dem Wert von \$offset zurück
\$wert=strrpos(\$str,\$such)	gibt die letzte Position von \$such in der Zeichenkette \$str zurück
\$erg=strstr(\$str,\$such)	sucht \$such in der Zeichenkette \$str und gibt die Teil-Zeichenkette von \$str ab der gefundenen Position bis zum Ende zurück
\$erg=substr(\$str,\$start,\$len)	gibt die Teil-Zeichenkette ab der Position \$start von \$str mit der Länge \$len zurück
\$erg=strcmp(\$str1,\$str2)	vergleicht \$str1 und \$str2 und gibt -1 zurück, wenn \$str1 < \$str2, 0 wenn beide Strings gleich sind und +1, wenn \$str1 > \$str2
\$erg=strcasecmp(\$str1,\$str2)	wie strcmp, berücksichtigt jedoch keine Groß- und Kleinschreibung

Funktionen auf Zeichenketten: Ersetzen von Zeichen

Funktion	Bedeutung
<code>\$erg=addslashes(\$str,\$charlist)</code>	setzt C-typische Escape-Zeichen vor jedem Sonderzeichen, dass in \$charlist angegeben ist
<code>\$erg=stripcslashes(\$str,\$charlist)</code>	entfernt C-typische Escape-Zeichen vor jedem Sonderzeichen, dass in \$charlist angegeben ist
<code>\$erg=addslashes(\$str)</code>	setzt einen Backslash vor speziellen Sonderzeichen
<code>\$erg=stripslashes(\$str)</code>	entfernt den gesetzten Backslash vor speziellen Sonderzeichen
<code>\$erg=ltrim(\$str)</code>	entfernt führende Leerzeichen
<code>\$erg=rtrim(\$str)</code>	entfernt nachfolgende Leerzeichen
<code>\$erg=trim(\$str)</code>	entfernt alle Leerzeichen am Anfang und Ende von \$str
<code>\$erg=str_replace(\$such,\$ers,\$str)</code>	ersetzt in \$str jedes Vorkommen von \$such durch \$ers

Funktionen auf Zeichenketten: Umwandlung

Funktion	Bedeutung
<code>\$erg=strrev(\$str)</code>	invertiert die Zeichenkette \$str
<code>\$erg=strtolower(\$str)</code>	wandelt \$str in Kleinbuchstaben um
<code>\$erg=strtoupper(\$str)</code>	wandelt \$str in Großbuchstaben um
<code>\$erg=ord(\$char)</code>	gibt den ASCII-Wert des Zeichens zurück
<code>\$char=chr(\$byte)</code>	gibt das Zeichen des eingegebenen ASCII-Wertes zurück
<code>\$arr=explode(\$sep, \$str)</code>	trennt \$str anhand von \$sep aus und liefert ein Datenfeld zurück
<code>\$str=implode(\$sep, \$arr)</code>	wandelt ein Datenfeld in eine Zeichenkette um und fügt zwischen den Elementen den Separator \$sep ein

Funktionen auf Zeichenketten: HTML-Bearbeitung

Funktion	Bedeutung
<code>\$erg=nl2br(\$str)</code>	wandelt Zeilenumbrüche in <code>
</code> um
<code>\$erg=htmlentities(\$str)</code>	konvertiert HTML-Zeichen, Umlaute und andere Sonderzeichen, um die Interpretation durch den Internet-Browser zu verhindern
<code>\$erg=rawurlencode(\$str)</code>	konvertiert Umlaute und Sonderzeichen einer Zeichenkette in Prozentzeichen und hexadezimalen ASCII-Wert zur Verwendung in einer URL
<code>\$erg=rawurldecode(\$str)</code>	macht die Konvertierung von <code>rawurlencode</code> rückgängig

Erzeugung von Ausgaben

Befehl	Beschreibung
echo str1 [, str2 ...]	Ausgabe von Strings
print(string)	Ausgabe eines Strings
printf(format [, args])	formatierte Ausgabe (entspricht C printf und Java System.out.printf)
<?= \$var ?>	Ausgabe einer Variable oder eines Ausdrucks (entspricht <?php print(\$var); ?>)
flush()	leert den Ausgabepuffer

Erzeugung von Ausgaben: Beispiel

```
<?php
```

```
$name = 'Frank';
```

```
echo "<b>", "Hallo ", $name, "</b>" , "<br />";
```

```
print("<b>" . "Hallo " . $name . "</b>" . "<br />");
```

```
printf("<b>Hallo %s</b><br />", $name);
```

```
?>
```

```
Hallo Frank  
Hallo Frank  
Hallo Frank
```

Einbinden externer Dateien

- Klassen oder Funktionen können aus anderen Dateien in eine PHP-Datei eingebunden werden per
 - `require_once("beispiel.php");` oder per
 - `include_once("beispiel.php");`
- Dies ist vergleichbar mit `#include` in C++.
- Eine bedingte Einbindung ist über eine `if`-Anweisung möglich.
- Auch der Dateiname kann zur Laufzeit berechnet werden.
- Jede Datei wird dabei nur einmal inkludiert.
- Eine fehlende Datei bewirkt
 - einen Abbruch bei `require_once` bzw.
 - eine Warnung bei `include_once`.
- Die Variablen darin sind übergreifend sichtbar!

Einbinden externer Dateien: Beispiel

```
<?php include_once('include/header.php'); ?>
```

```
<h2>Herzlich Willkommen!</h2>
```

```
<p>Diese Webseite informiert Sie über die neuesten Darts-Ergebnisse.</p>
```

```
<?php include_once('include/footer.php'); ?>
```

Einbinden externer Dateien: include/header.php & footer.php

- header.php:

```
<?php
    $wert=100; // auch alle PHP-Initialisierungen
?>
<html>
<head>
    <meta name="author" content="Frank Dopatka">
    <meta name="copyright" content="Frank Dopatka">
    <meta name="language" content="de">
    <title>Meine Homepage</title>
    <!-- JavaScript -->
    <!-- CSS -->
</head>
<body>
```

- footer.php:

```
<p>Copyright by Dr. Frank Dopatka, 2019</p>
</body>
</html>
```

Beenden eines Skripts

- Skripte können an beliebiger Stelle per Kommando unmittelbar beendet werden:
- `exit`;
beendet das Skript.
- `exit()`;
beendet das Skript.
- `exit($code)`;
beendet das Skript mit Fehlercode.
- `exit($string)`;
beendet das Skript und gibt den Fehlerstring aus.
- `die`;
ist ein Synonym für `exit`.

Zuweisungsoperatoren

Operation	Bedeutung
$z = x + y$	Addition von x und y in die Variable z
$z = x - y$	Subtraktion von x und y und Speicherung in die Variable z
$z = x * y$	Multiplikation von x und y und Speicherung in die Variable z
$z = x / y$	Division von x mit y und Speicherung in die Variable z
$z = x \% y$	Rest der Ganzzahldivision von x und y und Speicherung in die Variable z
$z = x . y$	Aneinanderreihen der Zeichenketten x und y und Speicherung in die Variable z

Kombinierte Zuweisungsoperatoren

Operation	Bedeutung
$\$x = \y	Inhalt der Variablen $\$y$ wird nach $\$x$ kopiert
$\$x = \&\y	Speicheradresse der Variablen $\$y$ wird auf die Speicheradresse der Variablen $\$x$ gesetzt
$\$x += \y	Addition von $\$y$ zur Variablen $\$x$
$\$x -= \y	Subtraktion von $\$y$ von der Variablen $\$x$
$\$x *= \y	Multiplikation von $\$y$ mit der Variablen $\$x$
$\$x /= \y	Division von $\$x$ mit $\$y$
$\$x \% = \y	Rest der Ganzzahldivision von $\$x$ und $\$y$
$\$x .= \y	Hinzufügen der Zeichenkette $\$y$ zu der Zeichenkette $\$x$

Vergleichsoperatoren

Operation	Bedeutung
$x > y$	liefert <i>TRUE</i> , wenn der Wert von x größer ist als der Wert von y
$x \geq y$	liefert <i>TRUE</i> , wenn der Wert von x größer oder gleich dem Wert von y ist
$x < y$	liefert <i>TRUE</i> , wenn der Wert von x kleiner ist als der Wert von y
$x \leq y$	liefert <i>TRUE</i> , wenn der Wert von x kleiner oder gleich dem Wert von y ist
$x == y$	liefert <i>TRUE</i> , wenn der Wert von x gleich dem Wert von y ist
$x != y$	liefert <i>TRUE</i> , wenn der Wert von x ungleich dem Wert von y ist
$x === y$	liefert <i>TRUE</i> , wenn der Wert von x gleich dem Wert von y ist und x und y vom selben Typ sind
$x !== y$	liefert <i>TRUE</i> , wenn der Wert von x ungleich dem Wert von y ist und die Datentypen von x und y unterschiedlich sind

Bit-Operatoren

Operation	Bedeutung
$x \& y$	binäre UND-Verknüpfung
$x y$	binäre ODER-Verknüpfung
$x \wedge y$	binäre XOR-Verknüpfung (liefert <i>TRUE</i> , wenn x oder y wahr ist, aber nicht beide)
$\sim x$	Komplement-Darstellung
$x \ll y$	verschiebt die Bits von x um y -Schritte nach links; jeder Schritt nach links bedeutet eine Multiplikation mit 2
$x \gg y$	verschiebt die Bits von x um y -Schritte nach rechts; jeder Schritt nach rechts bedeutet eine Division durch 2

Operatoren: Inkrement und Dekrement

Operation	Bedeutung
<code>++\$x</code>	erhöht den Wert von <code>\$x</code> um 1 und gibt den neuen Wert von <code>\$x</code> zurück
<code>\$x++</code>	gibt den Wert von <code>\$x</code> zurück und erhöht den Wert anschließend um 1
<code>--\$x</code>	verringert den Wert von <code>\$x</code> um 1 und gibt den neuen Wert von <code>\$x</code> zurück
<code>\$x--</code>	gibt den Wert von <code>\$x</code> zurück und vermindert den Wert anschließend um 1

Funktionen für Datum & Uhrzeit

Funktion	Bedeutung
<code>\$arr=gettimeofday()</code>	gibt die aktuelle Zeit in einem Datenfeld zurück
<code>\$erg=microtime()</code>	gibt den aktuellen UNIX-Zeitstempel in Mikrosekunden seit dem 01.01.1970, 00:00Uhr zurück
<code>\$erg=time(\$args)</code>	gibt den aktuellen UNIX-Zeitstempel sekundengenau in der Formatierung der Argumente zurück
<code>\$erg=mktime(\$std,\$min,\$sek, \$monat,\$tag,\$jahr)</code>	ermittelt den UNIX-Zeitstempel anhand der Zeitangabe

```
<?php
$arr = gettimeofday();
var_dump($arr);
?>
```

```
array(4) { ["sec"]=> int(1577391274) ["usec"]=> int(433227) ["minuteswest"]=> int(0) ["dsttime"]=> int(0) }
```

Funktionen für Datum & Uhrzeit

Funktion	Bedeutung
<code>\$erg=checkdate(\$monat,\$tag,\$jahr)</code>	überprüft eine Zeitangabe auf Gültigkeit unter Berücksichtigung der Schaltjahre und gibt einen Wahrheitswert zurück
<code>\$str=date(\$args)</code>	gibt das Datum in dem durch die Argumente gewünschten Format zurück
<code>\$arr=getdate(\$zeitstempel)</code>	gibt Informationen bezüglich des Datums im UNIX-Zeitstempel als Datenfeld zurück

Funktionen für Datum & Uhrzeit: Formatierung

Platzhalter	Bedeutung
A	AM oder PM
d	Tag des Monats mit führender Null
j	Tag des Monats ohne führende Null
D	abgekürzter Tag
l	vollständig ausgeschriebener Wochentag
F	vollständig ausgeschriebener Monat
m / n	Monat mit/ohne führender Null
M	abgekürzt geschriebener Monat
h / H	Stunde im 12/24-Stunden-Format mit führender Null
g / G	Stunde im 12/24-Stunden-Format ohne führende Null
i	Minuten mit führender Null
s	Sekunden mit führender Null
T	Anzahl der Tage des Monats
W	Wochentag als Zahl; 0 für Sonntag bis 6 für Samstag
y / Y	zweistellige vierstellige Jahresangabe
z	Tag im Jahr

```
echo(date("d.m.Y - H:i:s"));  
02.06.2018 - 22:54:16
```

mathematische Funktionen zur Konvertierung

Funktion	Bedeutung
<code>\$erg=decbin(\$var)</code>	konvertiert vom Dezimal-System ins Binär-System
<code>\$erg=bindec(\$var)</code>	konvertiert vom Binär-System ins Dezimal-System
<code>\$erg=dechex(\$var)</code>	konvertiert vom Dezimal-System ins Hexadezimal-System
<code>\$erg=hexdec(\$var)</code>	konvertiert vom Hexadezimal-System ins Dezimal-System
<code>\$erg=decoct(\$var)</code>	konvertiert vom Dezimal-System ins Oktal-System
<code>\$erg=octdec(\$var)</code>	konvertiert vom Oktal-System ins Dezimal-System
<code>\$erg=deg2rad(\$var)</code>	konvertiert Grad zum Bogenmaß
<code>\$erg=rad2deg(\$var)</code>	konvertiert Bogenmaß zu Grad
<code>\$erg=base_convert(\$var,\$base1,\$base2)</code>	konvertiert zwischen dem Zahlensystem <code>\$base1</code> in das Zahlensystem <code>\$base2</code>
<code>\$erg=floor(\$var)</code>	rundet eine Fließkommazahl auf die nächste Ganzzahl ab
<code>\$erg=ceil(\$var)</code>	rundet eine Fließkommazahl auf die nächste Ganzzahl auf
<code>\$erg=round(\$var)</code>	rundet einen Wert bei $\geq x.5$ auf und bei $< x.5$ ab

mathematische Funktionen für Zufallszahlen

Funktion	Bedeutung
<code>\$erg=getrandmax()</code>	ermittelt die höchstmögliche Zahl, die durch die Funktion rand erzeugt werden kann
<code>srand(\$var)</code>	legt über \$var einen internen Startwert für den Zufallsgenerator fest
<code>\$erg=rand(\$min,\$max)</code>	gibt eine Zufallszahl zwischen \$min und \$max zurück