PHP
erste Schritte

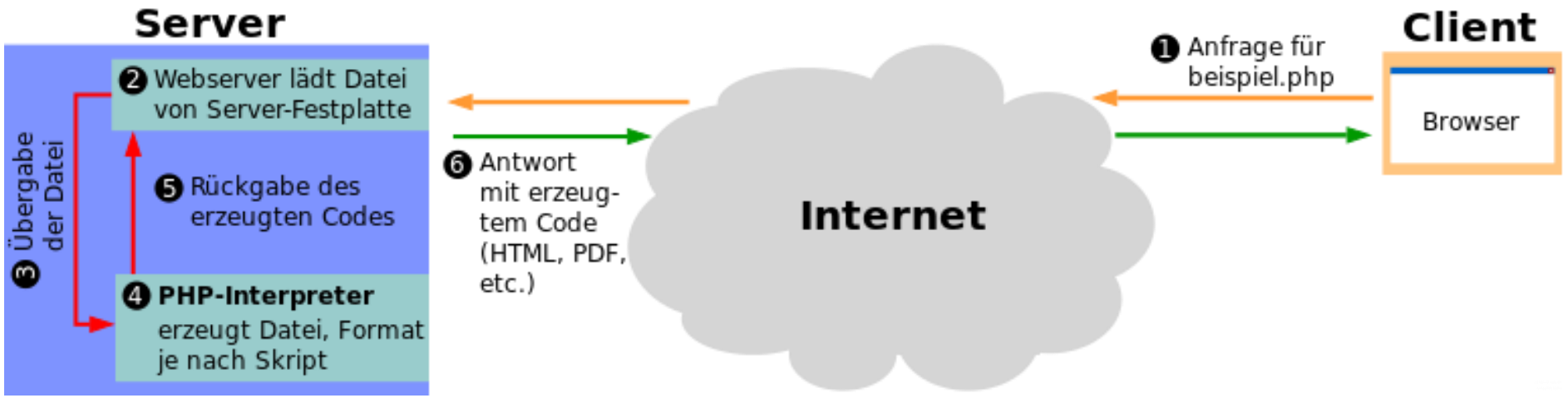
Was ist PHP?

- 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`**

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

echo KONSTANTENNAME;
```

- Ausgabe:

```
Ich bin der Inhalt
```

- Konstanten sind superglobal, d.h. überall 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>
```

Erster PHP-Code Clientseitig

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
<?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
<?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
<?php

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

?>
```

```
1
```

Variablen auslesen mit var_dump

```php
<?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
<?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 '<img src="portrait.jpg">';
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];
```

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

Funktionen auf Zeichenketten: Umwandlung

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

Funktionen auf Zeichenketten: HTML-Bearbeitung

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

```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.

Operatoren: 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

Operatoren: 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	Substraktion 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

Operatoren: Vergleichsoperatoren

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

Operatoren: Bit-Operatoren

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

Operatoren: Inkrement und Dekrement

Operation	Bedeutung
++$x	erhöht den Wert von $x um 1 und gibt den neuen Wert von $x zurück
$x++	gibt den Wert von $x zurück und erhöht den Wert anschließend um 1
--$x	verringert den Wert von $x um 1 und gibt den neuen Wert von $x zurück
$x--	gibt den Wert von $x zurück und vermindert den Wert anschließend um 1

Funktionen für Datum & Uhrzeit

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

```php
<?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
$erg=checkdate($monat,$tag,$jahr)	überprüft eine Zeitangabe auf Gültigkeit unter Berücksichtigung der Schaltjahre und gibt einen Wahrheitswert zurück
$str=date($args)	gibt das Datum in dem durch die Argumente gewünschten Format zurück
$arr=getdate($zeitstempel)	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: Konvertierungs-Funktionen

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

mathematische Funktionen: Funktionen für Zufallszahlen

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