PHP Basics

SAE Wien
Alexander Hofbauer
hofbauer.alexander@gmail.com

Geschichte

PHP steht rekursiv für “**PHP: Hypertext Preprocessor**”

1995: PHP/FI (“Personal Home Page Tools / Form Interpreter”)

1997: PHP/FI 2

1997: Neuentwicklung PHP 3

1999: Neuentwicklung des Kerns PHP 4

2004: PHP 5

2016: PHP 7.1

2021: **PHP 8** (v8.0.1 - 7. Jän. 2021)

http://php.net (!!)

+ Dokumentation
+ Diskussion, Meinungen, Tipps
+ Release Informationen

Eigenschaften von PHP

+ Interpretersprache
+ Anleihen aus C/C++, Perl, Unix Shell, Java
+ **serverseitig**
+ Vielzahl eingebauter Funktionen
+ Modular erweiterbar

Grundlagen

+ Dateiendung: `.php` (z.B. `index.php`)
+ PHP und HTML können gemischt werden → PHP-Tags

```php
<?php

// PHP Code

?>
```

Kommentare

```
+  // einzlg. Kommentar
+  # einzlg. Kommentar
+  /* mehrzlg.
Kommentar */
```

Nach Möglichkeit einheitlich kommentieren!

Best Practice:

```
/**
 * Kommentar
 */
```

GANZ GANZ WICHTIG!

...ganz wichtig!

In PHP wird nach jeder Anweisung genau ein **Semikolon** gesetzt;

Wird es vergessen → Syntaxfehler, das Skript wird nicht ausgeführt!

```
echo "hallo welt!";
```

!!

Variablen und Datentypen

Variablen

+ Beginnen mit einem Dollar-Zeichen: $
+ Gefolgt von einem Buchstaben oder Unterstrich (_)
+ Erlaubt sind Buchstaben und Ziffern (inkl. Umlaute ← nicht zu empfehlen)
+ Case-Sensitive! (Groß-/Kleinschreibung wird unterschieden)
+ Alphanumerisch & Unterstrich

Variablen

```php
<?php

$var = "Wir";
$Var = "lernen";
$vaR = "PHP";

echo "$var $Var $vaR!"; // Wir lernen PHP!

// Variablen Namen

$2birnen = "2 Birnen"; // Ungültig!
$_2birnen = "2 andere Birnen"; // GÜLTIG :D

?>
```

Datentypen

PHP wählt den Typ automatisch! ← Fluch & Segen (Verknüpfungen & Operationen)

Im Zweifelsfall immer manuell umwandeln.

```php
// Skalare Datentypen
$wahrheitswert = true; // boolean
$ganzzahl = 42;        // integer
$kommazahl = 3.1415;   // float (= double)
$zeichenkette = "foo"; // string

// Zusammengesetzte Datentypen
$feld = array('foo', 'bar'); // array
$objekt = new stdClass();    // object
$funktion = function () {    // callable
    echo "hello world";
};

// Spezielle Datentypen
$resource = fopen("foo", "w"); // resource
$nichts = NULL;                // NULL
```

Implizite Typumwandlung

z.B.: `$zahl = (int)$irgendeinWert;`

Dabei wird der Inhalt der Variable umgewandelt.

Typecasting passiert in PHP bei Bedarf automatisch:

```php
<?php

$foo = "0";             // $foo <-- string (ASCII 48)
$foo += 2;                // $foo <-- integer (2)
$foo += 1.3;            // $foo <-- float (3.3)
$foo = 5 + "10 Birnen"; // $foo <-- integer (15)

?>
```

Strings (Zeichenketten)

- aufeinanderfolgende Zeichen
- theoretisch beliebig lang
- Anführungszeichen
- **Einfache Anführungszeichen**: String wird **nicht** vom Compiler **interpretiert**
- **Doppelte Anführungszeichen**: String wird vom Compiler **interpretiert**

```php
$i = "Das ist ein String";
$j = 'Das hier auch :D';
```

```php
<?php

$i = ‘ein String‘;
$k = ‘Das ist $i‘;
echo $k; // Das ist $i

?>
```

```php
<?php

$i = ‘ein String‘;
$k = “Das ist $i”;
echo $k; // Das ist ein String

?>
```

Strings escapen

sollen Anführungszeichen in Strings verwendet werden, müssen diese “escaped” werden

```php
<?php

$i = ‘ein String‘;
$k = “Das ist \”$i\””;
echo $k; // Das ist “ein String”

?>
```

werden die Anführungszeichen gemischt verwendet, müssen sie nicht escped werden.

```php
<?php

$i = ‘ein String‘;
$k = “Das ist ‘$i’”;
echo $k; // Das ist ‘ein String’

?>
```

Strings escapen - Spezielle Zeichen

Escapen funktioniert nur in Strings mit doppelten Anführungszeichen!

```php
<?php

$zeilenumbruch  = "\n"; // "Line Feed"
$carriageReturn = "\r"; // Windows Zeilenumbruch: \r\n
$tabulator      = "\t";
$backslash      = "\\";
$singleQuote    = "\'";
$doubleQuote    = "\"";

?>
```

Zahlen

Integers

```php
<?php

$zahl = 42;       // decimal number
$zahl = -42;      // negative number
$zahl = 052;      // octal number
$zahl = 0x2A;     // hexadecimal number
$zahl = 0b101010;// binary number

?>
```

Floats/Doubles

```php
<?php

$zahl = 3.1415;
$zahl = -2.7182;

?>
```

Boolesche Werte

- logischer Wahrheitswert
- genau 2 mögliche Werte: **true** oder **false**

```php
<?php

$true = true;
$false = false;

?>
```

Arrays/Felder

+ dienen zum Abbilden von Reihen/Listen
+ Indizes sind mehr oder weniger beliebig

```php
<?php

// Array zuweisen
$farben[0] = ‘rot‘;
$farben[1] = ‘blau‘;
$farben[2] = ‘gelb‘;

// Oder
$farben = array(‘rot‘, ‘blau‘, ‘gelb‘);
echo $farben[1]; // blau

?>
```

Mehrdimensionale Arrays

```php
<?php

$plants = [
    ['oak', 'hickory', 'baobab'],
    ['tulip', 'rose', 'daffodile'],
    ['small shrubbery', 'burning shrubbery with spikes']
];

echo $plants[1][0]; // tulip

?>
```

Assoziative Arrays

```php
<?php

$plants = [
    'trees' => ['oak', 'hickory', 'baobab'],
    'flowers' => ['tulip', 'rose', 'daffodile'],
    'shrubberies' => ['small shrubbery', 'burning shrubbery with spikes']
];

echo $plants["flowers"][1]; // rose

?>
```

Datentypen prüfen

```
is_array();
is_bool();
is_double();
is_float();
is_integer();
is_numeric();
is_string();
is_null();
```

Konstanten

+ einmal definiert, können Konstanten nicht mehr verändert werden
+ gängige Konvention ist für Konstantennamen nur Großbuchstaben zu verwenden

```php
<?php

define('FOO', 'bar');
define('ANSWER_TO_THE_QUESTION_OF_LIFE_THE_UNIVERSE_AND_EVERYTHING', 42);

?>
```

Operatoren

Zuweisungsoperator

```php
// Einfache Zuweisung: =
$a = 37;

// Kombinierte Operatoren: +=, -=, *=, .=, /=, %=
$a += 5; // 42

$b = "Hallo";
$b .= ' i bims!';
echo $b; // Hallo i bims!
```

Vergleichsoperator

```
$a == $b  // true, wenn beide Werte gleich sind
$a != $b  // true, wenn beide Werte NICHT gleich sind

$a === $b // true, wenn beide Werte und Datentypen gleich sind
$a !== $b // true, wenn beide Werte und Datentypen NICHT gleich sind

$a > $b   // true, wenn linker Wert größer als rechter
$a < $b   // true, wenn linker Wert kleiner als rechter
$a >= $b  // true, wenn linker Wert größer als oder gleich dem rechten
$a <= $b  // true, wenn linker Wert kleiner als oder gleich dem rechten
```

Vergleichsoperator - Beispiele

```
42 == "42"        // true
"18" != 18.0      // false

42 === "42"       // false
"18" !== 18.0     // true

"42" > 41         // true
42 < "foobar"     // false
18 >= 18.0        // true
0b101010 <= "42" // true
```

Arithmetische Operatoren

```php
$a = 37;
$b = 5;

$c = $a + $b; // Addition --> 42
$c = $a - $b; // Subtraktion --> 32
$c = $a * $b; // Multiplikation --> 185
$c = $a / $b; // Division --> 7.4
$c = $a % $b; // Modulo --> 2
```

Inkrement/Dekrement Operatoren

```php
$a = 42;

++$a; // Prä-Inkrement; erhöht $a um 1 und gibt es zurück
$a++; // Post-Inkrement; gibt $a zurück und erhöht es dann um 1

--$a; // Prä-Dekrement
$a--; // Post-Dekrement
```

String Operatoren

```php
// Verketten: .
$a = "Hallo";
$b = "Welt";
$c = $a . ' ' . $b; // Hallo Welt

// Verketten und Zuweisen: .=
$a .= $b // HalloWelt

// Zeichenindizierung: []
echo $c[1]; // a
```

Logische Operatoren

```
$a && $b  // true, wenn beide Werte true sind
$a AND $b // true, wenn beide Werte true sind

$a || $b  // true, wenn mind. ein Wert true ist
$a OR $b  // true, wenn mind. ein Wert true ist

$a XOR $b // true, wenn beide unterschiedliche sind

!$a       // true, wenn der Wert false ist (Negation)
```

Referenzen

+ Referenzen verweisen auf denselben Inhalt im RAM

```php
<?php

$a = 1; // a: 1
$b = 2; // b: 2

$a = &$b; // a: 2

$a = 3; // a: 3, b: 3

?>
```

Ausgaben und Debugging

Ausgaben - echo

```php
<?php

echo "Hallo Welt";
echo ("Hallo”, “Welt");

$a = "Hallo Welt"; echo $a; // Hallo Welt

$b = "Welt"; echo "Hallo " . $b; // Hallo Welt

?>
```

Ausgaben - print

```php
<?php

print "Hallo Welt";
print ("Hallo Welt");

$a = "Hallo Welt"; print $a; // Hallo Welt

$b = "Welt"; print "Hallo " . $b; // Hallo Welt

?>
```

+ echo ist etwas schneller
+ nur ein Parameter möglich
+ print hat einen Rückgabewert 1

Ausgaben - printf

```php
<?php

$everything = "Allem";
$answer = 42;

$text = "Die Antwort auf die Frage nach dem Leben, dem Universum und %s lautet %d.";

printf($text, $everything, $answer);
// Die Antwort auf die Frage nach dem Leben, dem Universum und Allem lautet 42.

// %d = positive Ganzzahl
// %s = String

?>
```

Ausgaben - debugging: print_r

```php
<?php

$werte = ["foo", "bar"];

// rekursive Ausgabe
print_r($werte);



// Array
// (
//   [0] => foo
//   [1] => bar
// )

?>
```

Ausgaben - debugging: var_dump

```php
<?php

$werte = ["foo", "bar"];

// rekursive Ausgabe inkl. Länge & Typ
var_dump($werte);

// array(2) {
//   [0]=>
//   string(3) "foo"
//   [1]=>
//   string(3) "bar"
// }

?>
```

Kontrollstrukturen und Schleifen

Schleifen - for

```php
<?php

for (Initialisierung; Bedingung; Aenderungssequenz) {
    // Code
}


for ($i = 0; $i <= 100; $i++) {
    echo $i;
}

?>
```

+ Initialisierung
+ Bedingung
+ Änderung

Schleifen - while

```php
<?php

while (Bedingung) {
    // Code
}

while ($row = mysqli_fetch_assoc($res)) {
    echo $row['title'];
}

?>
```

+ Bedingung am Anfang
+ ist die Bedingung nicht erfüllt, wird die Schleife nicht durchlaufen

Schleifen - do/while

```php
<?php

do {
    // Code
} while (Bedingung)



$i = 0;
do {
    echo $i;
} while ($i > 0);

?>
```

+ Bedingung am Ende
+ wird mind. 1x durchlaufen

Schleifen - foreach

```php
<?php

foreach (Array as Wert) {
    // Anweisungen
}


foreach (Array as Key => Wert) {
    // Anweisungen
}

?>
```

+ Anweisungen können auf alle Elemente eines Arrays angewendet werden
+ kann **NUR** auf **Arrays** angewendet werden

Schleifen - foreach

Wert

```php
<?php

foreach ($array as $wert) {
    echo "Das Feld hat den Wert: $wert";
}

?>
```

Schlüssel & Wert

+ häufig für assoziative Arrays verwendet

```php
<?php

foreach ($array as $key => $wert) {
    echo "Index: $key, Wert: $wert";
}

?>
```

Bedingung - if/else

```php
<?php

if ($anzahl >= 15) {
    // something
} elseif ($anzahl == 10) {
    // something else
} else {
    // well, no condition was met -.-
}

?>
```

Bedingung - switch & match

```php
<?php // PHP 7

switch (8.0) {
  case '8.0':
    $result = "Oh no!";
    break;
  case 8.0:
    $result = "This is what I expected";
    break;
}
echo $result;
//> Oh no!

?>
```

+ Expression (Rückgabewert kann in eine Variable gespeichert werden)
+ “strict comparison”

```php
<?php // PHP 8

echo match (8.0) {
  '8.0' => "Oh no!",
  8.0 => "This is what I expected",
};
//> This is what I expected

?>
```

Funktionen und Parameter

Funktionen

```php
<?php

// Funktion definieren
function funktionsname (Parameter, Parameter, ...) {
    // Code
}

// Funktion aufrufen
funktionsname(Parameter);

?>
```

Funktionen - Type Hints & Union Types

```php
<?php

$a = 2;
$b = 3;

// “int|float” ist hier jeweils ein optionaler Typehint für Parameter und Rückgabewert
// (früher PHPDoc)
function calc(int|float $zahl1, int|float $zahl2): int|float {
    echo $zahl1 + $zahl2;
}

calc($a, $b);
?>
```

Funktionen - Parameter

+ Vorgabewerte für Parameter

```php
<?php

function func($text = "Whoop", $zahl = 3) {
    echo "$text $zahl";
}

func("other text", 42); // other text 42
func();                 // Whoop 3

?>
```

Funktionen - “Named Arguments”

+ Ermöglicht es einzelne optionale Parameter zu überspringen
+ Reihenfolge der Parameter ist nicht mehr wichtig

```php
<?php
// PHP 7
htmlspecialchars($string, ENT_COMPAT | ENT_HTML401, 'UTF-8', false);

// PHP 8
htmlspecialchars($string, double_encode: false);

?>
```

Funktionen - Parameter

```php
<?php

function appendString(&$string) {
    $string .= ' und ein angefuegter String';
}

$str = 'ein string';
appendString($str);
echo $str; // ein string und ein angefuegter
String

?>
```

+ Verweise als Parameter
+ normalerweise werden Änderungen an Variablen nur im aktuellen Gültigkeitsbereich (Scope) durchgeführt
  ← Verweise brechen diesen Scope auf