Interaktive Webseiten mit PHP und MySQL

Sommersemester 2003

Martin Ellermann
Heiko Holtkamp

Quellenhinweise

- Offizielle PHP-Webseite: http://www.php.net
- ZEND Technologies: http://www.zend.com
- Krause, Jörg: Programmieren lernen in PHP4. Hanser, 2001
- Krause, Jörg: PHP4 - Grundlagen und Profiwissen. Hanser, 2001, 2. Aufl.
- Greenspan, Jay; Bulger, Brad: MySQL/PHP-Datenbankanwendungen. MITP, 2001
- Naber, Daniel: Lebendiges Web - Programmiersprachen für dynamische Webseiten. c't 8/2003 (7.4.2003), S. 92 ff.
- Reeg, Christoph; Hatlak, Jens: Datenbank, MySQL und PHP. 2002. http://www.reeg.net.

Interaktive/Dynamische Webseiten

- Webserver können nicht nur starre Inhalte anzeigen, sondern HTML-Seiten auch anhand von Benutzereingaben zur Laufzeit individuell anfertigen.

- Es gibt zwei verschiedene Konzepte für dynamische Webseiten, die unterschiedliche Einsatzgebiete und Eigenschaften haben.

Konzepte dynamischer Webseiten

- Beim ersten Konzept generiert ein Webserver für einen bestimmten Benutzer, z.B. aufgrund einer Anfrage, eine individuelle HTML-Seite. Dabei laufen Programme, meist in Skriptsprachen wie Perl oder PHP geschrieben, auf dem Webserver.
  => Server-seitige Web-Programmiersprachen

- Beim zweiten Konzept ist die Webseite selbst interaktiv, beispielsweise in Form einer kleinen Anwendung oder eines Menüs, das bei berühren mit der Maus aufklappt. Innerhalb solcher HTML-Seiten kommen meist JavaScript und Java-Applets zum Einsatz.
  => Client-seitige Web-Programmiersprachen

Client-seitige Web-Programmiersprachen

- Java (Java-Applets)
- JavaScript
  - Die Kombination von HTML, CSS und JavaScript wird als *Dynamic HTML* bezeichnet.

Server-seitige Web-Programmiersprachen

- CGI in Verbindung mit
  - Perl/CGI
  - C/C++
  - Fortran
  - TCL
  - Python
  - etc.
- PHP
- Perl (Embperl)
- JSP (Java Server Pages)
- etc.

Dynamische Webseiten mit PHP

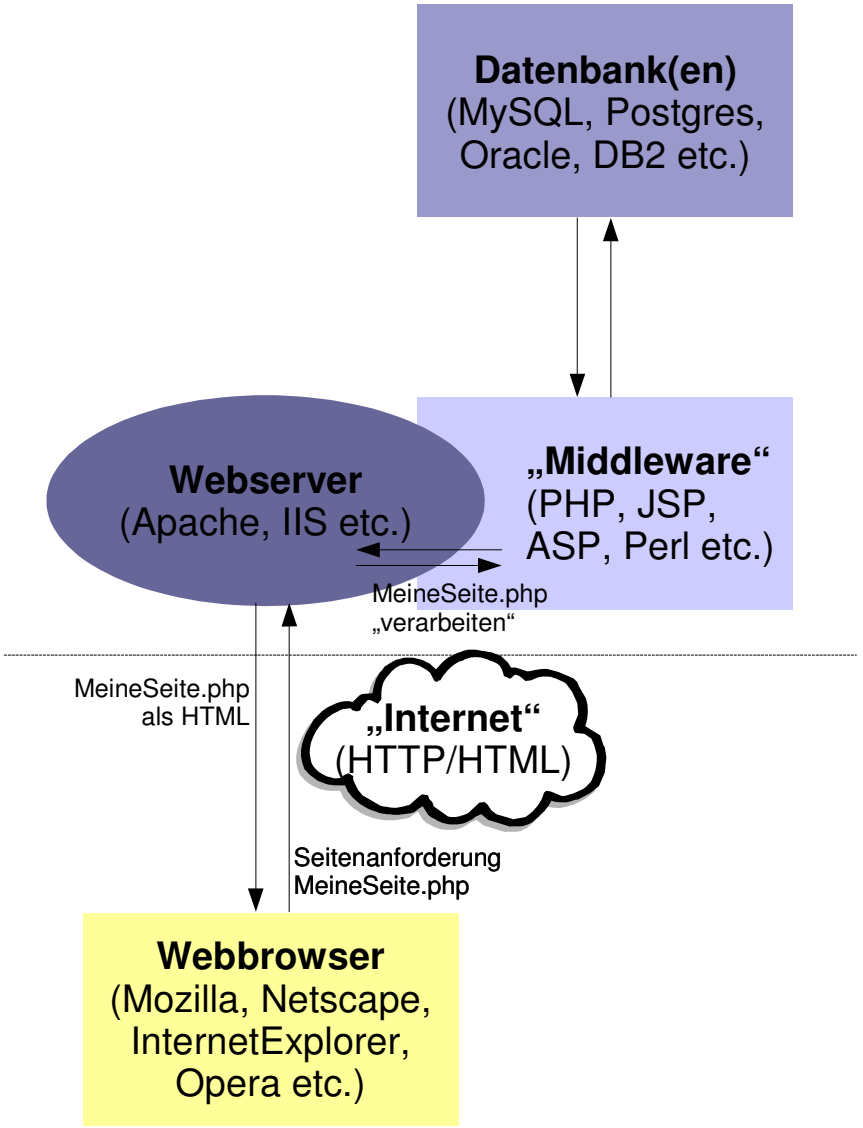
- PHP ist eine serverseitige in HTML eingebettete Skriptsprache
- Das Akronym PHP steht für *PHP Hypertext Preprocessor*
- PHP ist speziell für auf Web-Programmierung ausgerichtet.
- Der Programmcode wird in die HTML-Quelldatei geschrieben und somit i.A. nicht in einer extra „PHP-Datei“ abgelegt.
  - Der Webserver entscheidet allerdings anhand der Endung (Extension) einer Datei, ob sie durch den PHP-Interpreter geschickt wird oder nicht. Üblich für HTML-Dateien mit PHP-Code sind die Endungen .php, .php3, .php4 oder .phtml.
- HTML- und Programmlogik lassen sich beliebig mischen.

Kurze Geschichte von PHP (Quelle: PHP-Manual unter www.php.net)

- Frühling 1994 die erste Version von PHP, entwickelt von *Rasmus Lerdorf* als *Personal Home Page Tools.*
- Mitte 1995 neue Version PHP/FI Ver. 2. FI steht für *Form Interpreter,* ein weiteres Paket, welches Lerdorf für die Verarbeitung von HTML-Formulardaten geschrieben hat.
- Bis 1997 „Boom“ von PHP/FI mit weit über 50.000 Webseiten, die PHP/FI einsetzen.
- Mitte 1997 Änderung bei der Weiterentwicklung von PHP. Um PHP herum entsteht ein Entwicklerteam mit den Hauptpersonen *Zeev Suraski* und *Andi Gutmans*.
- 1997/1998 neuer Parser für PHP, dies führt zur Version *PHP3*.
- 1998 Gründung von *ZEND Technologies* durch Suraski und Gutmans.
- Ende 2000/Anfang 2001 *PHP4*. Die aktuelle Version vewendet die sogenannte *Zend Scripting Engine* für den Parser und ist als Modul für fast alle gängigen Webserver verfügbar (PHP3 nur für den Apache-Webserver). PHP4 wird zu einem System, das auch zunehmend im kommerziellen Bereich Anwendung findet.

Dynamische Webseiten mit PHP

- Beim Aufruf einer PHP-Seite liest der Webserver die Datei, führt den eingebetteten PHP-Code aus und schreibt dessen Ausgabe in die HTML-Seite.



PHP und seine Syntax

- Damit der Server zwischen HTML und PHP-Code unterscheiden kann, müssen die PHP-Code-Abschnitte speziell gekennzeichnet werden, d.h. mit einem Tag kenntlich gemacht (‚Escaping from HTML‘):
  - `<? /* PHP-Code */ ?>`
  - `<?php /* PHP-Code */ ?>`
  - `<SCRIPT LANGUAGE=„PHP“> /* PHP-Code */ </SCRIPT>`

- PHP-Tags dürfen mehrfach in einer Datei mit HTML-Code vorkommen und gemischt werden.
- Der Programmzusammenhang geht durch die Teilung nicht verloren.

‚Advanced Escaping‘

- PHP bietet eine alternative Syntax für IF-Abfragen an, mit der ganze HTML-Blöcke anhand der Bedingungen in der Abfrage ausgegeben werden können:

```
<?php IF ($a = $b): ?>
    <b>a ist gleich b</b>
<?php ELSE : ?>
    <b>a ist ungleich b</b>
<?php ENDIF; ?>
```

PHP und seine Syntax

- Kommentare im PHP-Code werden zwischen `/*` und `*/` gestellt.
- Einzeilige Kommentare oder Kommentare am Ende einer Zeile werden mit `//` (oder `#`) eingeleitet.
- Jeder PHP-Befehl schließt mit einem Semikolon „`;`“ ab.

- PHP ist stark von C, aber auch von Java und Perl (die ihrerseits von C beeinflusst wurden) geprägt.

Ein erstes Beispiel

Hello World!

Variablen (Teil 1)

- Eine Variable in PHP beginnt mit einem Dollarzeichen ‚$', gefolgt vom Namen der Variablen.
- Variablennamen sind *casesensitiv*, d.h. es wird zwischen Groß- und Kleinschreibung unterschieden.
- Gültige Variablennamen beginnen mit einem Buchstaben oder einem Unterstrich (*underscore*), gefolgt von einer beliebigen Anzahl Buchstaben*, Zahlen oder Unterstrichen.
  (* Buchstabe heißt hier a-z, A-Z und die ASCII-Zeichen von 127 bis 255)

```
$var = "Bob";
$Var = "Joe";
echo "$var, $Var";          // gibt "Bob, Joe" aus

$4site  = 'geht nicht';     // ungültig, beginnt mit einer Zahl
$_4site = 'geht';  // gültig, beginnt mit einem Unterstrich
$gültig = 'geht auch';      // gültig, 'ü' ist ein erlaubtes ASCII-Zeichen
```

Variablen (Teil 1)

- In PHP werden Variablen nicht deklariert, also nicht wie in anderen Hochsprachen am Anfang eines Blockes definiert und einem Typ zugeordnet.
- Der Typ einer Variablen wird normalerweise nicht vom Programmierer bestimmt, sondern zur Laufzeit von PHP, abhängig vom Kontext in der die Variable genutzt wird.
- Wohl aber kann der Programmierer den Typ einer Variablen auch bestimmten: *type casting*.
- Diese Eigenart erfordert vom Programmierer Disziplin! Ein falsch getippter Variablenname führt so z.B. nicht nicht zu einer Fehlermeldung (kann wohl aber zu einem Fehler führen!).

Variablen (Teil 1)

```
<?php
  $zahl1 = 100;
  $zahl2 = 50;
  $ergebnis = $Zahl1 - $zahl2;
  echo $ergebnis;
?>
```

Ausgabe: -50

Der *echo*-Befehl

- Der echo-Befehl gibt einen oder mehrere Strings aus
  `echo ( string arg1 [, string argn...])`
- echo() ist keine Funktion, sondern ein internes Sprachkonstrukt. Aus diesem Grund brauchen beim echo-Befehl keine Klammern angegeben werden.

Typen

- PHP unterstützt 8 primitive Typen.
  - Vier skalare Typen
    - Boolean
    - Integer
    - Fließkomma-Zahl (float)
    - String / Zeichenkette
  - Zwei zusammengesetzte Typen
    - Array
    - Object
  - Zwei spezielle Typen
    - Resource
    - NULL
- Anmerkung: In Referenzen zu PHP-Funktionen werden oft *mixed*-Parameter angegeben. Dieser Pseudo-Typ weist darauf hin, dass es mehrere Möglichkeiten für diesen Parameter gibt.

Typen

- Der Typ einer Variablen wird normalerweise nicht vom Programmierer bestimmt. Zur Laufzeit von PHP wird entschieden, welchen Typs eine Variable ist, abhängig vom Zusammenhang in dem die Variable benutzt wird.
- Soll die Umwandlung in einen bestimmten Variablen-Typ erzwungen werden, kann dies über Typ-Zuweisungen (*type casting*) gemacht werden.
- Eine Variable kann sich in bestimmten Situationen unterschiedlich "verhalten", abhängig vom Typ dem die Variable zu dem Zeitpunkt entspricht (Typ-Veränderung, *type juggling*).

Typen - Boolean

- Der Typ Boolean ist der einfachste Typ.
- Ein Boolean drückt einen Wahrheitswert aus.
- Ein Boolean kann `TRUE` oder `FALSE` sein. Die Schlüsselwörter sind case-insensitive (`True` = `TRUE` = `true` ...)
- Achtung: Der Boolean-Typ wurde erst mit PHP4 eingeführt!
- Bei der Typ-Konvertierung werden die folgenden Werte als `FALSE` angenommen:
  - das Boolean FALSE
  - die Integer-Zahl 0 (Null)
  - die Fließkomma-Zahl 0.0 (Null)
  - die leere Zeichenkette und die Zeichenkette "0"
  - ein Array ohne Elemente
  - ein Objekt ohne Elemente
  - der spezielle Typ NULL (einschließlich nicht definierter Variablen)
- Jeder andere Wert wird als `TRUE` angesehen

Typen - Integer

- Ein Integer ist eine Nummer aus der Menge Z = {..., -2, -1, 0, 1, 2, ...}.
- Ganzzahlen können in dezimaler (10-basierter), hexadezimaler (16-basierter, `0x`) oder oktaler (8-basierter, `0`) Schreibweise angegeben werden, wahlweise mit vorangestelltem Vorzeichen (- oder +).

```
$a = 1234; // Dezimalzahl
$a = -123; // eine negative Zahl
$a = 0123; // Oktalzahl (entspricht 83 dezimal)
$a = 0x1A; // Hexadezimalzahl (entspricht 26 dezimal)
```

- Die Größe eines Integer-Wertes ist plattformabhängig!
- PHP unterstützt keine vorzeichenlosen Integer-Werte.

Typen - Integer

- Zahlen und Ergebnisse von Operationen mit Integern die jenseits der Grenzen des Typs Integer liegen, werden als Typ Float interpretiert.
- Um einen Wert ausdrücklich nach Integer zu konvertieren kann die Typ-Umwandlung mittels (int) oder (integer) genutzt werden.
- In den allermeisten Fällen ist es jedoch nicht notwendig die Umwandlung selbst vorzunehmen. Ein Wert wird automatisch konvertiert, falls ein Operator, eine Funktion oder eine Kontrollstruktur ein Integer Argument erfordert.

Typen - Floats

- Fließkommazahlenwerte ("floats", "doubles" oder "reelle Zahlen") können durch eine der folgenden Anweisungen zugewiesen werden:

```
$a = 1.234;
$a = 1.2e3;
$a = 7E-10;
```

- Die Größe einer Fließkommazahl ist plattformabhängig, dennoch stellt ein Maximum von ~1.8e308 mit einer Genauigkeit von 14 Nachkomma-Stellen einen üblichen Wert dar (das entspricht dem 64-Bit IEEE-Format).
- Neben dem Typ Float hat PHP keine weiteren Typen für Fließkommazahlen (wie z.B. C mit Float, Double und Long Double)

Typen – Strings / Zeichenketten

- Ein String ist eine Folge von Zeichen. In PHP entspricht ein Zeichen einem Byte, das heißt, dass exakt 256 unterschiedliche Zeichen möglich sind.
- Anmerkung: Für einen String stellt die Länge kein Problem dar. Von PHP-Seite aus gibt es keine praktische Grenze für die Größe eines Strings. Daher gibt es keinen Grund sich Sorgen über lange Strings zu machen.
- Ein String kann auf drei verschiedene Weisen geschrieben werden.
  - Einfache Anführungszeichen (single quoted)
  - Doppelte Anführungszeichen (double quoted)
  - Heredoc Syntax

Typen – Strings / Zeichenketten

- Single Quoted
  - Der einfachste Weg einen String zu schreiben ist, ihn in einfache Anführungszeichen zu schreiben (‘).
  - Um ein einfaches Anführungszeichen direkt auszugeben, muss dieses mit einem Backslash (\) „escaped“ werden.
  - Um einen Backslash direkt auszugeben, muss dieser mit einem Backslash (\) „escaped“ werden.

```
echo 'Das ist ein einfacher String';
echo 'Arnold sagte einmal: "I\'ll be back"';
echo 'Sind Sie sicher, dass Sie C:\\*.* löschen
wollen?';

$var = "nur zum Vergnügen";
echo 'Das mache ich $var';
```

Typen – Strings / Zeichenketten

- Double Quoted
  - Wenn ein String in doppelten Anführungszeichen (") eingeschlossen ist, versteht PHP mehr Escape-Sequenzen für spezielle Zeichen:

Sequenz	**Bedeutung**
\n	Zeilenvorschub (LF oder 0x0A als ASCII-Code)
\r	Wagenrücklauf (CR oder 0x0D als ASCII-Code)
\t	horizontaler Tabulator (HT oder 0x09 als ASCII-Code)
\\	Backslash / Rückstrich
\$	Dollar-Symbol
\"	doppelte Anführungszeichen

  - Der wichtigste Vorteil von double-quoted Strings ist die Tatsache, dass Variablennamen ausgewertet werden.

Typen – Strings / Zeichenketten

- Variablenanalyse (variable parsing)
  - Wird ein String in doppelten Anführungszeichen oder mit heredoc angegeben, werden enthaltene Variablen ausgewertet (geparst).
  - Es gibt zwei Syntax-Typen, eine *einfache* und eine *komplexe*
    - Die einfache Syntax ist die geläufigste und bequemste. Sie bietet die Möglichkeit eine Variable, einen Array-Wert oder eine Objekt-Eigenschaft auszuwerten (parsen).
    - Die komplexe Syntax wurde in PHP 4 eingeführt und ist an den geschweiften Klammern {}erkennbar, die den Ausdruck umschließen.
  - Sobald ein Dollarzeichen ($) auftaucht, wird der Parser versuchen, gierig so viele Zeichen wie möglich zu bekommen, um einen gültigen Variablennamen zu bilden.

Typen – Strings / Zeichenketten

- Sobald ein String als numerischer Wert ausgewertet wird, wird der resultierende Wert und Typ wie folgt festgelegt:
  - Der String wird als Float ausgewertet, wenn er eines der Zeichen '.', 'e' oder 'E' enthält.
  - Ansonsten wird er als Integer-Wert interpretiert.
- Der Wert wird durch den Anfangsteil des Strings bestimmt. Sofern der String mit gültigen numerischen Daten beginnt, werden diese als Wert benutzt. Andernfalls wird der Wert 0 (Null) sein.

```
$foo = 1 + "10.5";                // $foo ist float   (11.5)
$foo = 1 + "-1.3e3";              // $foo ist float   (-1299)
$foo = 1 + "bob-1.3e3";           // $foo ist integer (1)
$foo = 1 + "bob3";                // $foo ist integer (1)
$foo = 1 + "10 Kleine Schweine"; // $foo ist integer (11)
$foo = 4 + "10.2 Ferkel";         // $foo ist float   (14.2)
$foo = "10 Schweine " + 1;        // $foo ist integer (11)
$foo = "10.0 Schweine " + 1;      // $foo ist float   (11)
$foo = "10.0 Schweine " + 1.0;    // $foo ist float   (11)
```

Typen - Arrays

- Ein Array in PHP ist eine *geordnete Abbildung*. Eine Abbildung ist ein Typ der Werte auf Schlüssel abbildet.
- Ein Array muss nicht definiert werden, da der PHP-Interpreter den Speicher dynamisch verwaltet.
- PHP beherrscht *indizierte* (d.h. der Index ist ein Integer) und auch *assoziative* (d.h. der Index ist ein String) Arrays. Vermischungen sind zulässig.

Typen - Arrays

- Syntax von Arrays
  - Ein Array kann über das Sprachkonstrukt `array()` erstellt werden: es erwartet eine Anzahl von per Komma separierten Schlüssel/Wert-Paaren (`key => value`)

```
$a = array('color' => 'red',
           'taste' => 'sweet',
           'shape' => 'round',
           'name' => 'apple');
```

  - Ein Array kann auch durch explizites Zuweisen von Werten verändern werden `$arr[key] = value;`

```
$a['color'] = 'red';
$a['taste'] = 'sweet';
$a['shape'] = 'round';
$a['name' = 'apple';
```

    Existierte das Array $arr noch nicht, wird es mit der ersten Zuweisung eines Wertes erzeugt.

Typen - Arrays

- Mehrdimensionale Arrays
  - Da der Typ eines Array-Elements beliebig sein kann, kann das Element auch wiederum ein Array sein:

```
$mitarbeiter = array("m1" => array("Ellermann", "Martin"),
                     "m2" => array("Holtkamp", "Heiko")
                     );
```

oder

```
$mitarbeiter["m1"][0] = "Ellermann";
$mitarbeiter["m1"][1] = "Martin";
$mitarbeiter["m2"][0] = "Holtkamp";
$mitarbeiter["m2"][1] = "Heiko";
```

Typen - Arrays

- Es sollten immer Anführungszeichen für einen assoziativen Index eines Arrays benutzt werden. Zum Beispiel sollte `$foo['bar']` und nicht `$foo[bar]` geschrieben werden.
- **Beispiel:**
```
$foo[bar] = 'besser nicht';
echo $foo[bar];
```
- Es ist falsch, funktioniert aber. Warum ist es dann falsch? Der Grund ist, dass dieser Code eine undefinierte Konstante (bar) enthält, anstatt eines Strings. Es funktioniert, weil die undefinierte Konstante in einen String mit gleichem Namen umgewandelt wird.

Typ-Konvertierung (type juggling)

- PHP erfordert (bzw. unterstützt) keine explizite Typ-Definitionen bei der Deklaration von Variablen.
- Der Typ einer Variablen wird durch den Zusammenhang bestimmt in dem die Variable benutzt wird. Das bedeutet, dass bei der Zuweisung einer Zeichenkette/eines Strings zu einer Variablen var diese Variable zum Typ String wird.
- Ein Beispiel für die automatische Typ-Konvertierung ist der Additionsoperator '+'.

```
$foo = "0";                    // $foo ist vom Typ String (ASCII 48)
$foo += 2;                     // $foo ist jetzt vom Typ Integer (2)
$foo = $foo + 1.3;             // $foo ist nun vom Typ float (3.3)
$foo = 5 + "10 Kleine Ferkel";   // $foo ist vom Typ Integer (15)
$foo = 5 + "10 Kleine Schweine"; // $foo ist vom Typ Integer (15)
```

Typ-Konvertierung (type casting)

- Explizite Typ-Umwandlung in PHP funktioniert vielfach wie in C: Der Name des gewünschten Typs wird vor der umzuwandelnden Variablen in Klammern gesetzt.
- Folgende Umwandlungen sind möglich:
  - (int), (integer) - nach integer
  - (bool), (boolean) - nach boolean
  - (float), (double), (real) - nach float
  - (string) - nach string
  - (array) - nach array
  - (object) - Wandlung zum Objekt
- Beispiel:

```
$foo = 10;              // $foo ist ein Integer
$bar = (float) $foo     // $bar ist ein Float
```

Konstanten

- In PHP gibt es neben Variablen auch Konstanten.
- Bei Konstanten ist, wie der Name schon sagt, der Wert nicht veränderlich.
- Konstanten haben einen globalen Gültigkeitsbereich.
- Konstantennamen sind *casesensitiv*, d.h. es wird zwischen Groß- und Kleinschreibung unterschieden.
- Gültige Konstantennamen beginnen mit einem Buchstaben oder einem Unterstrich (*underscore*), gefolgt von einer beliebigen Anzahl Buchstaben*, Zahlen oder Unterstrichen. (* Buchstabe heißt hier a-z, A-Z und die ASCII-Zeichen von 127 bis 255)

Konstanten

- Eine Konstante wird über die Funktion define() definiert. Einmal definiert, kann eine Konstante weder verändert noch gelöscht werden.
- Konstanten können nur skalare Datentypen (Boolean, Integer, Float und String) enthalten.
- Den Wert einer Konstanten erhält man durch die Angabe ihres Namens.

```
$var = "variabler Wert";
define ("CONSTANT", "konstanter Wert");
echo $var;
$var = CONSTANT;
echo $var;
$CONSTANT = "Problem?";
echo Constant;
```

Operatoren – Arithmetische Operatoren

Beispiel	Name	Ergebnis
$a + $b	Addition	Summe von $a und $b
$a - $b	Subtraktion	Differenz zwischen $a und $b
$a * $b	Multiplikation	Produkt von $a und $b.
$a / $b	Division	Quotient von $a und $b
$a % $b	Modulus	Rest von $a geteilt durch $b

- Der Divisions-Operator ("/") gibt immer eine Fließkommazahl zurück, auch wenn die zwei Operanden Ganzzahlen sind (oder Zeichenketten, die nach Ganzzahlen umgewandelt wurden).

Operatoren - Zuweisungsoperatoren

- Der einfachste Zuweisungsoperator ist "=".
- Der Zuweisungsoperator bedeutet, dass dem linken Operanden der Wert des Ausdrucks auf der rechten Seite zugewiesen wird ("wird gesetzt auf den Wert von" und nicht wie oft gesagt wird "ist gleich").
- Der Wert eines Zuweisungsausdruckes ist der zugewiesene Wert. D.h. der Wert des Ausdruckes "$a = 3" ist 3. Das erlaubt es, einige raffinierte Dinge anzustellen:

```
$a = ($b = 4) + 5
/* $a ist nun gleich 9 und $b wurde auf den
Wert 4 gesetzt. */
```

Operatoren - Zuweisungsoperatoren

- Zusätzlich zu dem oben vorgestellten Zuweisungsoperator "=" gibt es "kombinierte Operatoren" für alle binären, arithmetischen und String-Operatoren, die es erlauben, den Wert einer Variablen in einem Ausdruck zu benutzen, und dieser anschließend das Ergebnis des Ausdrucks als neuen Wert zuzuweisen.

```
$a = 3;
$a += 5;
/* setzt $a auf den Wert 8, als ob wir geschrieben
haetten: $a = $a + 5; */
$b = "Hallo ";
$b .= "Du!";
/* setzt $b auf den Wert "Hallo Du!", aequivalent zu $b
= $b . "Du!"; */
```