



Web Backend



Inhoud

- Inleiding
- Syntax
- Anatomie php.net
- Afdrukken in een document
- Variables
- Operatoren
- Conditional statements
- Arrays

Inhoud

- Array functions
- Looping statements
- Functions
- \$_GET
- \$_POST
- Controle \$_GET/\$_POST



Inleiding

- Wat is PHP
 - Scripttaal (<-> programmeertaal)
 - 1994 IBM ([Rasmus Lerdorf](#))
 - Perl / C / Python
 - **P**ersonal **H**ome **P**age (vroeger)
 - **PHP**: **H**ypertext **P**reprocessor (recursief acroniem)
 - Server side
 - Alternatieven:



Inleiding

- Doel van php:
 - Dynamisch informatie verwerken (oa.).
 - HTML = statisch
 - Onderhoudsintensief
 - PHP = dynamisch
 - Manier (=script) om het onderhoud van HTML/inhoud te vereenvoudigen.

Inleiding

- Multiplatform (Windows / Linux / Mac / ...)

- Enkele 'All-in-One'-pakketten

- LAMP (**L**inux **A**pache **M**ySQL **P**HP)
- MAMP (**M**acintosh **A**pache **M**ySQL **P**HP)
- WAMP (**W**indows **A**pache **M**ySQL **P**HP)
- XAMPP (**X** = Linux, Windows, Mac en Solaris

Apache

MySQL

PHP

Perl)



Inleiding

- Compatibel met meest gangbare servers (Apache, ISS, ...)
- Meerdere database-types ondersteunen
 - MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, MongoDB, ...
- Gratis te downloaden & gebruiken
- Makkelijk om te leren

Syntax

- Wat is een PHP-bestand?
 - Extensie: .php (of .php3 of .phtml)
 - Inhoud:
 - PHP-script
 - HTML/JavaScript/CSS
 - geschreven als string binnen <?php ?>-tags
 - standaard notatie buiten <?php ?>-tags
 - Output kan getoond worden als:
 - HTML, JavaScript, CSS, ...
 - JPEG, PNG, GIF, ...
 - eender welke bestandsextensie

Syntax

- Script block
 - Start met `<?php` en eindigt met `?>`
 - Elke opdrachtregel wordt afgesloten met een `;` (!!!)
 - Behalve de laatste regel die voor de closing `?>`-tag komt
- 2 manieren om PHP te integreren
 - Inline PHP
 - HTML-document met script-blokken (vb. [voorbeeld-syntax-inline-php](#))
 - Full PHP
 - PHP-script dat alle output regelt (vb. [voorbeeld-syntax-full-php](#))

Syntax

- Beste manier om PHP te integreren
 - scheiden van logica en output
 - variabelen bovenaan tussen scripttags definiëren
 - Onder scripttag -> HTML, CSS, JS
 - via script tags variabelen in HTML invullen

Syntax

- Commentaar in PHP script-blokken (vb. [voorbeeld-syntax-commentaar](#))
 - `// Dit is commentaar`
 - `/* Dit
is
comentaar op
meerdere regels */`
 - `# Dit is ook commentaar`
- Alle informatie over de versie van PHP en alle parafenalia:
 - `phpinfo()` (vb. [voorbeeld-syntax-phpinfo](#))

Syntax

- Vragen over PHP? Eén adres!

<http://www.php.net>

Anatomie php.net



What is PHP?

PHP is a widely-used general purpose scripting language that is especially suited for Web development and can be embedded into HTML.

If you are new to PHP and want to get some idea of how it works, try the [introduction tutorial](#). After that, check out the online [manual](#), and the [example archive](#) sites and some of the other resources available in the [links section](#).

We wondered how popular PHP is? See the [statistics page](#).

Thanks To

- [easypHP](#)
- [Dmitry](#)
- [Jan Nebecek](#)
- [Sergey Canalis](#)
- [Michael Sullivan](#)
- [Sera JETI-MAXING](#)
- [CGI Open Source Lab](#)
- [Yuhao Jin](#)
- [NEX-CSS.NET](#)
- [Rajasekhar](#)
- [Robbert](#)
- [Schroeder Webhosting](#)
- [Radip Limpi](#)
- [Facebook](#)
- [Scott@CZ.com](#)
- [SourceGuru](#)
- [Brett - Krish Gorden](#)

Related sites

- [Apache](#)
- [MySQL](#)
- [PostgreSQL](#)
- [Zend Technologies](#)

Connectivity

Upcoming conferences: [DevConf 2012](#) [Oulu PHP Conference 2012](#)

Calling for papers: [NorthEast PHP conference](#)

PHP 5.4.3 and PHP 5.3.13 Released!

[14 Nov 2012] The PHP development team would like to announce the immediate availability of PHP 5.4.3 and PHP 5.3.13. All users are encouraged to upgrade to PHP 5.4.3 or PHP 5.3.13.

The releases complete a fix for a [vulnerability](#) in CGI-based setups (CVE-2012-2311). Note: mod_php and php-fpm are not vulnerable to this attack.

PHP 5.4.3 fixes a buffer overflow vulnerability in the [apache_request_header\(\)](#) [CVE-2012-2329]. The PHP 5.3 series is not vulnerable to this issue.

For source downloads of PHP 5.4.3 and PHP 5.3.13 please visit our [downloads page](#). Windows binaries can be found on [windows.php.net/downloads](#). The list of changes are recorded in the [Changelogs](#).

PHP 5.3.12 and 5.4.2 and the CGI flaw (CVE-2012-1823)

[08 Nov 2012] PHP 5.3.12/5.4.2 do not fix all variations of the CGI issue described in CVE-2012-1823. It has also come to our attention that some sites use an insecure wrapper script to run PHP. These scripts will use \$* instead of "\$@" to pass parameters to php-cgi which causes a number of issues. Again, people using mod_php or php-fpm are not affected.

One way to address these CGI issues is to reject the request if the query string contains a "&" and no "?". It can be done using Apache's mod_rewrite like this:

```

RewriteCond %{QUERY_STRING} "[&]*" [OR]
RewriteCond %{QUERY_STRING} "!^?" [NC]
RewriteRule .* - [F,L]
    
```

Note that this will block otherwise safe requests like ?p=40 so if you have query parameters that look like that, adjust your regex accordingly.

Another set of releases are planned for Tuesday, May, 8th. These releases will fix the CGI flaw and another CGI-related issue in apache_request_header [5.4 only]. We apologize for the inconvenience created with these releases and the lack of communication around them.

PHP 5.3.12 and PHP 5.4.2 Released!

[08 Nov 2012] There is a vulnerability in certain CGI-based setups (**Apache+mod_php and nginx+php-fpm are not affected**) that has gone unnoticed for at least 8 years. [Section 7 of the CGI spec](#) states:

Some systems support a method for supplying a [sic] array of strings to the CGI script. This is only used in the case of an "indexed" query. This is identified by a "GET" or "HEAD" HTTP Request with a URI search string not containing any unencoded "&" characters.

Sic, requests that do not have a "?" in the query string are treated differently from those who do in some CGI implementations. For PHP this means that a request containing ? is may dump the PHP source code for the page, but a request that has ?&= is fine.

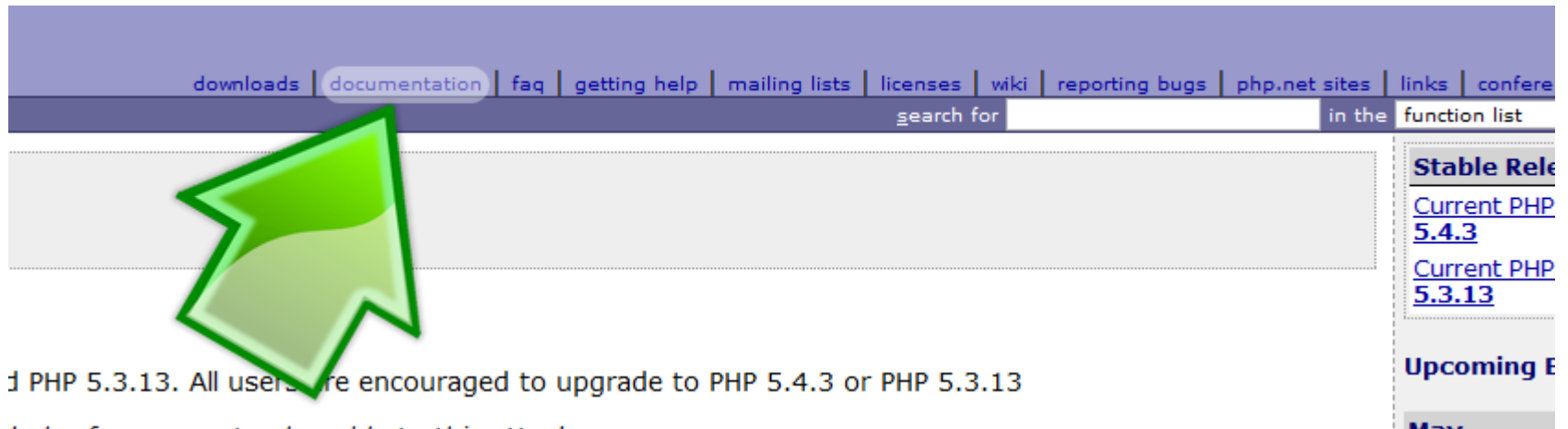
A large number of sites run PHP as either an Apache module through mod_php or using php-fpm under nginx. Neither of these setups are vulnerable to this. Straight shebang-style CGI also does not appear to be vulnerable.

If you are using Apache mod_cgi to run PHP you may be vulnerable. To see if you are, just add ?& to the end of any of your URIs. If you see your source code, you are vulnerable. If your site renders normally, you are not.

To fix this, update to PHP 5.3.12 or PHP 5.4.2.

Anatomie php.net

- Documentation (= volledig handleiding)



Anatomie php.net

- Zoeken in de documentation



Anatomie php.net

- Zoekresultaten

Related snippet found for 'function'

PHP contains thousands of functions. You might be interested in how a [function is defined](#), or [how to read a function prototype](#).




PHP Function List

function doesn't exist. Closest matches:

[function exists](#)
[create function](#)
[runkit function redefine](#)
[apd dump function table](#)
[reflectionfunctionabstract](#)
[register tick function](#)
[bcompiler write function](#)

[badfunctioncallexception](#)
[runkit function add](#)
[runkit function remove](#)
[override function](#)
[get defined functions](#)
[sqlite create function](#)
[bcompiler write functions from file](#)

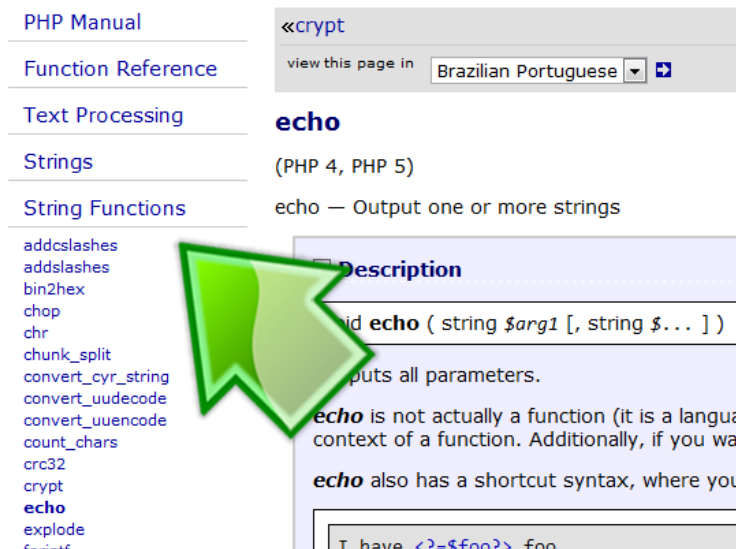
Site Search Results

-  [Functions - Manual](#)
Note about function names: -- According to the specified regular expression ([a-zA-Z_\x7f-\xff]*), a function name like this one
[php.net/function](#) - 4 May 2012 - [Cached](#)
-  [User-defined functions - Manual](#)
User-defined functions. A function may be defined using syntax such as the following:
[php.net/manual/en/functions.user-defined.php](#) - 8 May 2012 - [Cached](#)
-  [addslashes - Manual](#)



Anatomie php.net

- Analyse manual (vb. functie "echo")
 - Plaats van zoekterm binnen de manual



The screenshot shows the PHP Manual interface. On the left is a sidebar with a list of functions under the 'String Functions' category. A green arrow points to the 'echo' entry in this list. The main content area on the right displays the details for the 'echo' function, including its description and usage examples.

PHP Manual

Function Reference

Text Processing

Strings

String Functions

- addslashes
- addslashes
- bin2hex
- chop
- chr
- chunk_split
- convert_cyr_string
- convert_uuencode
- convert_uuencode
- count_chars
- crc32
- crypt
- echo**
- explode
- ...

«crypt

view this page in Brazilian Portuguese

echo

(PHP 4, PHP 5)

echo — Output one or more strings

Description

void **echo** (string *\$arg1* [, string *\$...*])

puts all parameters.

echo is not actually a function (it is a language construct). Additionally, if you want to use it in a function context, you can use the **echo** function.


echo also has a shortcut syntax, where you can write:

```
I have <?=$foo> foo
```

Anatomie php.net

— Description

[«crypt](#)

view this page in Brazilian Portuguese 

echo

(PHP 4, PHP 5)

echo — Output one or more strings

Description

```
void echo ( string $arg1 [, string $... ] )
```

Outputs all parameters.

echo is not actually a function (it is a language construct), so you are not required to use pa context of a function. Additionally, if you want to pass more than one parameter to **echo**, th

echo also has a shortcut syntax, where you can immediately follow the opening tag with an

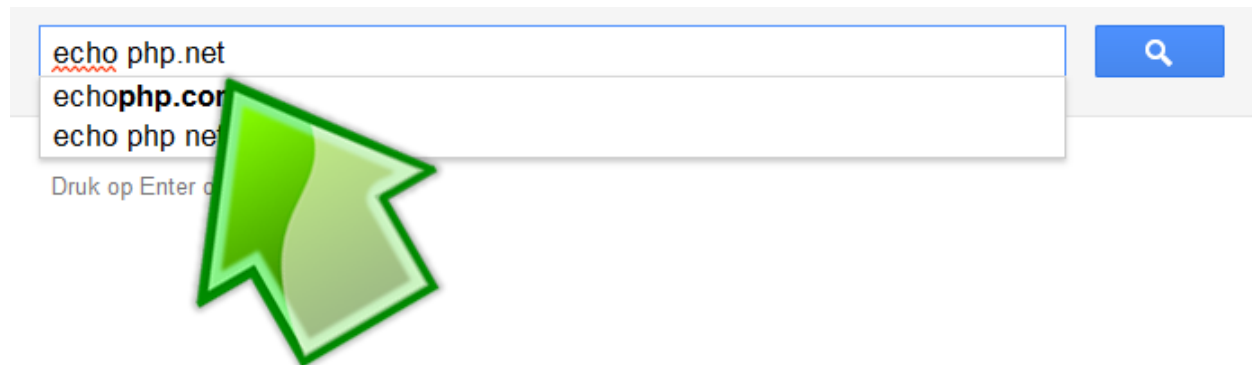
```
I have <?=$foo?> foo.
```

Anatomie php.net

- Parameters
 - Wat moet meegegeven worden
- Return value
 - Wat de functie teruggeeft
- Examples
- Notes
- See Also
- Comments
 - staan veel kant en klare oplossingen voor veel voorkomende problemen

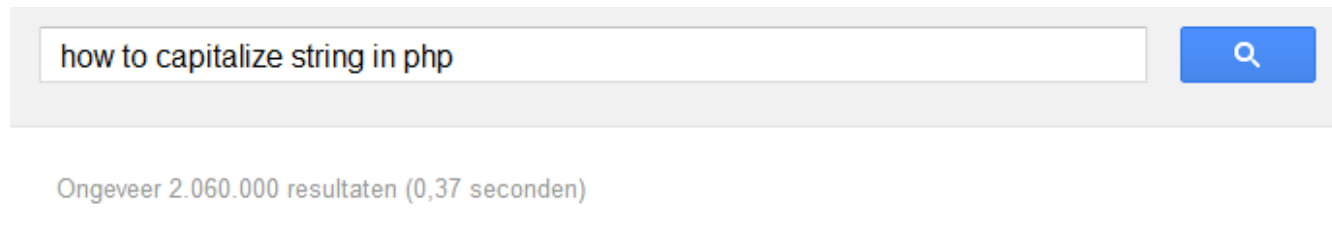
Anatomie php.net

- **Tip:** handiger zoeken d.m.v. Google
 - Je weet waar je naar op zoek bent
 - Zoekterm = functienaam + php.net



Anatomie php.net

- Je weet niet naar wat je op zoek bent
 - Zoekterm = Engelse vraag + in php



A screenshot of a search interface. It features a light gray background. At the top, there is a white search input field containing the text "how to capitalize string in php". To the right of the input field is a blue button with a white magnifying glass icon. Below the input field, the text "Ongeveer 2.060.000 resultaten (0,37 seconden)" is displayed in a smaller, gray font. A thin horizontal line is positioned below this text.

- Betrouwbaar?

Anatomie php.net

- Websites **php.net** & **stackoverflow.com** = goed!

[PHP Tutorial - Capitalization](#)

www.tizag.com/phpT/php-string-strtoupper-strtol... - Vertaal deze pagina

PHP - String Capitalization Functions. If you've ever wanted to manipulate the **capitalization** of your **PHP strings**, then this lesson will be quite helpful to you.

[ucfirst Make a string's first character uppercase - PHP](#)

php.net/manual/en/function.ucfirst.php - Vertaal deze pagina

string ucfirst (**string** \$str). Returns a **string** with the first character of str **capitalized**, if that character is alphabetic. Note that 'alphabetic' is determined by the ...

[strtoupper Make a string uppercase - PHP](#)

php.net/manual/en/function.strtoupper.php - Vertaal deze pagina

string strtoupper (**string** \$string). Returns **string** with all alphabetic characters converted to **uppercase**. Note that 'alphabetic' is determined by the current locale.

[ucwords Uppercase the first character of each word in a string - PHP](#)

php.net/manual/en/function.ucwords.php - Vertaal deze pagina

Returns a **string** with the first character of each word in str **capitalized**, if that ... (Note: it is advised to use a recent version of **PHP** when implementing this function ...)



Anatomie php.net

- Wees kritisch/analyseer website
 - » Datum?
 - » Commentaren?
 - » Reclame?
- Het eerste zoekresultaat is niet altijd wat je zoekt
- Neem de tijd om de eerste paar zinnen van elk zoekresultaat te lezen => antwoord zit hier vaak al in vervat

Afdrukken naar het scherm

– echo

- Enkel voor strings!
- Gebruik `<?= $var ?>` om dingen in html af te drukken
 - Bv `<p>Hallo, ik heet <?= $naam ?></p>`
 - Gebruik deze notatie énkél om variabelen af te drukken. Laat tekst die niet veranderd moet worden in de HTML staan
 - Dus zeker niet:
~~`<p><?= 'Hallo, ik heet' . $naam ?></p>`~~
 - Let op: niet elke server heeft deze shorthand echo standaard ingeschakeld (php.ini: short_open_tag = on)
 - (vb. [voorbeeld-afdrukken-echo](#))

Afdrukken naar het scherm

– **var_dump()**

- Dient enkel om te debuggen -> nooit in productieomgeving
- Geeft extra informatie over variabele (string/boolean/array/...) -
> vergemakkelijkt debuggen
- Standaard onoverzichtelijke opmaak
 - Oplossen door **print_r()** in samenwerking met <pre>-element
 - » Omslachtig!
 - XDEBUG kan helpen!
- (vb. [voorbeeld-afdrukken-var-dump](#))

Afdrukken naar het scherm

- XDEBUG
 - Module die debuggen in PHP aangenamer kan maken
 - » Enorm uitgebreid en krachtig -> best is samenwerking met uitgebreidere IDE (Eclipse, Netbeans, ...)
 - Inschakelen in XAMPP:
 - » C:\xampp\php\php.ini
 - Zoeken naar XDEBUG
 - Alle ; verwijderen onder deze XDEBUG-codeblock
 - Een ; in de php.ini file betekent een lijn in commentaar
 - » XAMPP/Apache heropstarten
 - » **(voor mac-gebruikers:**
http://docs.joomla.org/Edit_PHP.INI_File_for_XDebug)

Afdrukken naar het scherm


- Opdracht
 - opdracht-comments


- Wat zijn variables?
 - Vergelijkbaar met algebra:

$$x = 5$$

Variables

X = **5**

 **variable**

 **value**

$$\underbrace{x}_{\text{variable}} = \underbrace{5 + y}_{\text{expression}}$$

- Variable in php
 - Start altijd met een **\$**-teken gevolgd door de **naam van de variable**
 - De variable name moet **beginnen met een letter** OF een **underscore** (cijfers zijn NIET toegestaan)
 - De variable name kan **enkel letters, cijfers of een underscore** (**_**) bevatten

- Variable in php

- De variable name mag geen spaties bevatten
- De variable name is case sensitive (A ≠ a)
- Een type declaration is niet noodzakelijk (= loosely typed)
- De variable wordt gecreëerd wanneer er een value aan toegekend wordt

- Schrijfwijze

- Voor een variable die een string (= stuk tekst) bevat

\$myDrink = "Cuba Libre";

\$myDrink = 'Cuba Libre';

- Voor een variable die énkél een cijfer bevat

\$myGrade = 20;

Variables

- Schrijfwijze

- Voor een variable die niets bevat

\$containerVariable = null;

-> kan zowel een string, integer of ... zijn

\$containerVariable = "";

-> kan enkel een string zijn

- Variables kunnen ook boolean, array, double, object, resource, of "unknown type" zijn (later meer)

Variables

– Strings

- Een string is een stuk tekst

`$albumTitle = "Lullabies to paralyze";`

(vb. `voorbeeld-variables-string`)

- Verschillende strings kan men samenvoegen =

CONCATENATION

- Gebeurt door middel van een `.` (vb. `voorbeeld-variables-concatenation`)

Variables

– Strings

- Probleem met quotes (vb. [voorbeeld-variables-quote-problem](#))

- `$feeling = 'I'm fine';`

- Oplossing

- `$feeling = 'I\'m fine';`

- `$feeling = "I'm fine";`

Variables

– Stringfuncties

- **strlen()** (vb. [voorbeeld-variables-string-strlen-fn](#))
 - Retourneert de lengte van een string
- **strpos()** (vb. [voorbeeld-variables-string-strpos-fn](#))
 - Kijkt of een string voorkomt in een andere string
 - Indien deze string teruggevonden wordt => retourneert positie van string
 - Zoniet: retournt *false*

Variables

– Data types

- boolean: TRUE of *FALSE* (best altijd in hoofdletters)

`$a = FALSE;` (geeft als waarde 0 of niets)

`$a = TRUE;` (geeft als waarde 1)

- integer: geheel getal (wordt niet tussen quotes gezet)

`$a = 1234;` // decimaal

`$a = -123;` // negatief decimaal

`$a = 0123;` // octaal

`$a = 0x1A;` // hexadecimaal

- double: getal met cijfers na het punt (GEEN komma!)

`$a = 1.234;` // decimaal

- string

`$animal = 'paard';` of `$animal = "paard";`

Variables

— Data types

(later)

- array
- object
- resource
- null
- unknown type

Variables

- Strings (corrigeren)
 - opdracht-verbeter
- Strings (concatenate)
 - opdracht-string-concatenate
- Strings (string functions)
 - opdracht-string-extra-functions

Operators

Arithmetic Operators	Beschrijving	Voorbeeld	Resultaat
+	add	<code>\$x = 3;</code> <code>\$x = \$x + 4;</code>	7
-	subtract	<code>\$x = 5;</code> <code>\$x = 8 - \$x;</code>	3
*	multiply	<code>\$x = 3;</code> <code>\$x = \$x * 2;</code>	6
/	divide	<code>\$x = 6;</code> <code>\$x = \$x / 2;</code>	3
%	modulo	<code>\$x = 10;</code> <code>\$x = \$x % 4;</code> <code>\$x = \$x % 5;</code> <code>\$x = \$x % 3;</code>	2 (rest) 0 1 (rest)
++	increment	<code>\$x = 3;</code> <code>\$x++;</code>	4
--	decrement	<code>\$x = 3;</code> <code>\$x--;</code>	2

(vb. [voorbeeld-operators-arithmetic](#))

Operators

Assignment operators	voorbeeld	Resultaat
=	\$x = 3;	3
+=	\$x = 3; \$x += 5;	$(3 + 5) = 8$
-=	\$x = 3; \$x -= 1;	$(3 - 1) = 2$
*=	\$x = 3; \$x *= 2;	$(3 * 2) = 6$
/=	\$x = 6; \$x /= 2;	$(6 / 2) = 3$
.=	\$x = 3; \$x .= 2;	$(3 . 2) = 32$
%=	\$x = 8; \$x %= 4;	$(8 \% 2) = 0$

OPM: waar mogelijk, kies assignment boven de arithmetic operators (= overzichtelijker!)

Operators

Comparison operators	beschrijving	voorbeeld	resultaat
==	Is gelijk aan	"test" == "test" "4" == 4	TRUE TRUE
===	Is gelijk aan (incl. data type)	"4" === 4	FALSE
!=	Is niet gelijk aan	4 != 3	TRUE
>	Is groter dan	5 > 9	FALSE
<	Is kleiner dan	3 < 9	TRUE
>=	Is groter dan of gelijk aan	8 >= 9	FALSE
<=	Is kleiner dan of gelijk aan	8 <= 8	TRUE

(vb. [voorbeeld-operators-comparison](#))



Operators

Logical operators	beschrijving	voorbeeld	resultaat
&&	and	<code>\$x = 4;</code> <code>\$y = 8;</code> <code>(\$x < 10 && \$y > 1)</code>	TRUE
	or	<code>\$x = 4;</code> <code>\$y = 8;</code> <code>(\$x == 2 \$y == 3)</code>	FALSE
!	is not	<code>\$x = 4;</code> <code>\$y = 8;</code> <code>!(\$x == \$y)</code>	TRUE

- OPGELET:

~~`if ($x > 5 && < 10) { ... }`~~ => **FOUT**

`if ($x > 5 && $x < 10) { ... }` => **CORRECT**

Conditional statements

- vier verschillende conditional statements:
 - if statement
 - if ... else statement
 - if ... elseif... else statement
 - switch statement
- Functie: code uitvoeren wanneer er voldaan wordt aan een of meerdere voorwaarden

Conditional statements

- if statement

- Syntax:

```
if ( condition )
```

```
{
```

```
    uit te voeren code;
```

```
}
```

Conditional statements

- shorthand if statement

- Syntax:

- `(condition) ? code bij true : code bij false;`

- Wordt vooral gebruikt om op basis van een bepaalde condition een string of int aan een variabele toe te kennen.
 - niet voor het uitvoeren van grote stukken code

- (vb. `voorbeeld-conditional-statements-if`)

- Opdracht: `opdracht-conditional-statements`

Conditional statements

- if ... else statement

- Syntax:

```
if (condition)
{
    uit te voeren code;
}
else
{
    uit te voeren code;
}
```

(vb. [voorbeeld-conditional-statements-if-else](#))

- Opdracht: [opdracht-if-else](#)

Conditional statements

- if ... elseif ... else statement

- Syntax:

```
if (condition)
{
    uit te voeren code;
}
elseif (condition)
{
    uit te voeren code;
}
else
{
    uit te voeren code;
}
```

(vb. [voorbeeld-conditional-statements-if-elseif](#))

- Opdracht: [opdracht-if-else-if](#)

Conditional statements

- Switch statement
 - Dient enkel om één variabele te vergelijken met meerdere waarden
 - Syntax:

```
switch ($variablename)
{
    case value1:
        uit te voeren code;
        break;
    case value2:
        uit te voeren code;
        break;
    default:
        uit te voeren code;
}
```

(vb. [voorbeeld-conditional-statements-switch](#))

Conditional statements

- Switch statement

- Operatoren zijn niet toegestaan in de case condition!
- **break;** is noodzakelijk (anders wordt alles eronder ook uitgevoerd)
- Default is de actie die uitgevoerd wordt als er aan geen enkele conditie wordt voldaan.
- Opdracht: opdracht-switch


Arrays

- Een array is een variable die meerdere values kan bevatten.
- Er zijn twee soorten arrays
 - Numeric array (**begint altijd bij 0**)
 - `$frisdrank[0] = 'cola';`
 - Associative array
 - `$frisdrank['type'] = 'zero';`

Arrays

- Syntax

\$array[0] = 'fanta';


arrayname key value

Arrays

- Doel van array: values samenbundelen

– Bv. `$fri sdrank1 = ' Col a' ;`
 `$fri sdrank2 = ' Fanta' ;`
 `$fri sdrank3 = ' Spa' ;`

Beter:

`$fri sdrank[0] = ' Col a' ;`
`$fri sdrank[1] = ' Fanta' ;`
`$fri sdrank[2] = ' Spa' ;`

Arrays

- Array samenstellen: numeric array

- `$fri sdrank = array(' Col a' , ' Fanta' , ' Spa');`

OF

- `$fri sdrank[] = ' Col a' ;`
`$fri sdrank[] = ' Fanta' ;`
`$fri sdrank[] = ' Spa' ;`

Arrays

- Array samenstellen: associative Array

- `$frisdrank = array('Cola' => 'Zero', 'Fanta' => 'Regular');`

OF

- `$frisdrank['Cola'] = 'Zero';`
`$frisdrank['Fanta'] = 'Fanta';`

Arrays

- Inhoud van een array bekijken
 - Specifieke array value:
 - Numeric: `$fri sdrank[2] ;`
 - Associative: `$fri sdrank[' Col a'] ;`
 - De volledige array-inhoud:
 - `var_dump($fri sdrank) ;`
 - `print_r($fri sdrank) ;` (gebruik `<pre>...</pre>`)
 - (vb. `voorbeeld-arrays-opbouw`)

Arrays

- Multidimensional array
 - Array in een array
 - Multidimensional array maken:
 - `$frisdrank = array('Cola' => array('Regular', 'Zero', 'Light'), 'Fanta' => array('Regular', 'Lemon'));`
 - OF
 - `$frisdrank['Cola'] = array('Regular', 'Zero', 'Light');`
`$frisdrank['Fanta'] = array('Regular', 'Lemon');`
 - (vb. `voorbeeld-arrays-multidimensioneel`)

Arrays

- Opdracht: [opdracht-arrays-basis](#)

Array functions

- `count()`
 - Telt het aantal waarden van een array
 - Retourneert het aantal waarden (retourneert 0 als er geen waarden zijn)
 - (vb. `voorbeeld-arrays-function-count`)

Array functions

- `in_array()`
 - Kijkt of een waarde in een array voorkomt
 - Retourneert TRUE indien teruggevonden, FALSE indien niet
 - (vb. [voorbeeld-arrays-function-in-array](#))

Array functions

- Verdere array functies?
 - (vb. [voorbeeld-arrays-function-extra](#))
 - ...
 - [Php.net](#)
- Opdracht
 - [opdracht-array-functies](#)

Looping statements

- Looping statement voert een code block uit gedurende en herhaalt dit code block tot er aan een bepaalde conditie wordt voldaan.
- Vier soorten looping statements:
 - while loop
 - do... while loop
 - for loop
 - foreach loop

Looping statements

- While loop
 - Voert een code block uit zolang er aan de conditie wordt voldaan
 - Syntax:

```
while (condi ti on)
{
    uit te voeren code;
}
```

(vb. [voorbeeld-looping-statements-while](#))

- Opdracht: [opdracht-while](#)

Looping statements

- do... while loop
 - Het verschil met de while loop is dat hier de code block eerst wordt uitgevoerd en daarna pas wordt gekeken of er aan een bepaalde conditie wordt voldaan alvorens de code weer uit te voeren.
 - Syntax:

```
do
{
    uit te voeren code;
}
while ( condition )
```

(vb. [voorbeeld-looping-statements-do-while](#))

Looping statements

- for loop
 - Voert een code block een vooropgesteld aantal keer uit.
 - Syntax:

```
for (i n i t; c o n d i t i o n; i n c r e m e n t) {  
    u i t   t e   v o e r e n   c o d e;  
}
```

(vb. [voorbeeld-looping-statements-for](#))
- Opdracht: [opdracht-for](#)

Looping statements

- foreach loop

- Loopt doorheen een array

- Syntax:

```
foreach ($array as $value) {  
    uit te voeren code die gebruik maakt van $value;  
}
```

(vb. [voorbeeld-looping-statements-foreach-value](#))

Looping statements

- foreach loop

```
foreach ($array as $key => $value) {  
    uit te voeren code die gebruik maakt van $key en  
    $value;  
}
```

OPM: de naam van de parameters \$key en \$value mag je volledig zelf kiezen.

(vb. [voorbeeld-looping-statements-foreach-key-value](#))

- Opdracht: [opdracht-foreach](#)

Looping statements

- Alternatieve syntax voor looping statements
 - if/for/foreach/while/switch hebben alternatieve schrijfwijze

```
<?php if ($a == 5): ?>  
    A is equal to 5  
<?php endif; ?>
```

- Na elke conditie een :
- Om de looping statement af te sluiten:
endif/endfor/endforeach/endwhile/endswitch

Looping statements

- Alternatieve syntax voor looping statements
 - **DOEL:** overzichtelijk php mengen in HTML
 - Maak hier gebruik van!
 - Nog altijd niet overzichtelijk -> beste slechtste oplossing
 - Alternatieven
 - Templating languages (Blade, Markdown, ...)
 - [Template animation](#)
 - (vb. **voorbeeld-looping-statements-alternative-syntax**)

Functions

- Wanneer bepaalde berekeningen vaak terugkomen is het best hiervoor een functie te gebruiken.
- Een functie definiëren

```
function functionName($parameter)
{
    uit te voeren code;
}
```

(vb. voorbeeld-functions-basis)

Functions

- Functie met return definiëren (definiëren)
 - functie voert bewerking uit met bepaald resultaat.
 - Dit resultaat kan men teruggeven d.m.v. return
 - Bij een return wordt de functie beëindigd. Alle code erna die binnen dezelfde functie staat, wordt niet meer uitgevoerd.

```
function functionName($parameter)
{
    uit te voeren code;
    return $resultaat;
}
```

(vb. [voorbeeld-functions-return](#))

Functions

- Syntax
 - Een function begint altijd met function gevolgd door de function name.
 - Function name begint altijd met een letter of een underscore (**NOOIT** een getal!)
 - Function name moet een duidelijke betekenis hebben (**NOOIT** afkortingen!)
 - Variabelen die met de function name worden 'ingesteld' heten **parameters**

Functions

- Hoe een functie callen (aanroepen)?

- Zonder arguments:

- `functionName()`

- Met één argument

- `functionName($argument);`

- Met meerdere arguments

- `functionName($argument01, 'argument02value');`

- (vb. [voorbeeld-functions-arguments-parameters](#))

Functions

- Opdracht: opdracht-functies

Functions

- Scope van variabelen
 - De reden waarom er in functies met parameters wordt gewerkt: het bereik van variabelen.
 - Variabelen zijn niet overal aanspreekbaar
 - Er is een verschil tussen **global** en **local variables**
 - **Global**: variabele beschikbaar in heel het document
 - **Local**: enkel beschikbaar binnen een functie
 - (vb. **voorbeeld-functions-scope**)

Functions

- Scope

- Vanuit een functie een globale variabele aanspreken:

- define de variable in de function eerst door middel van volgende syntax:

global \$variable;

(vb. **voorbeeld-functions-global**) (OPM: \$GLOBALS['variablename'])

Functions

- Scope

- Een functie een local variable laten onthouden

- define de variable in de function door middel van volgende syntax:

Static \$variable = value;

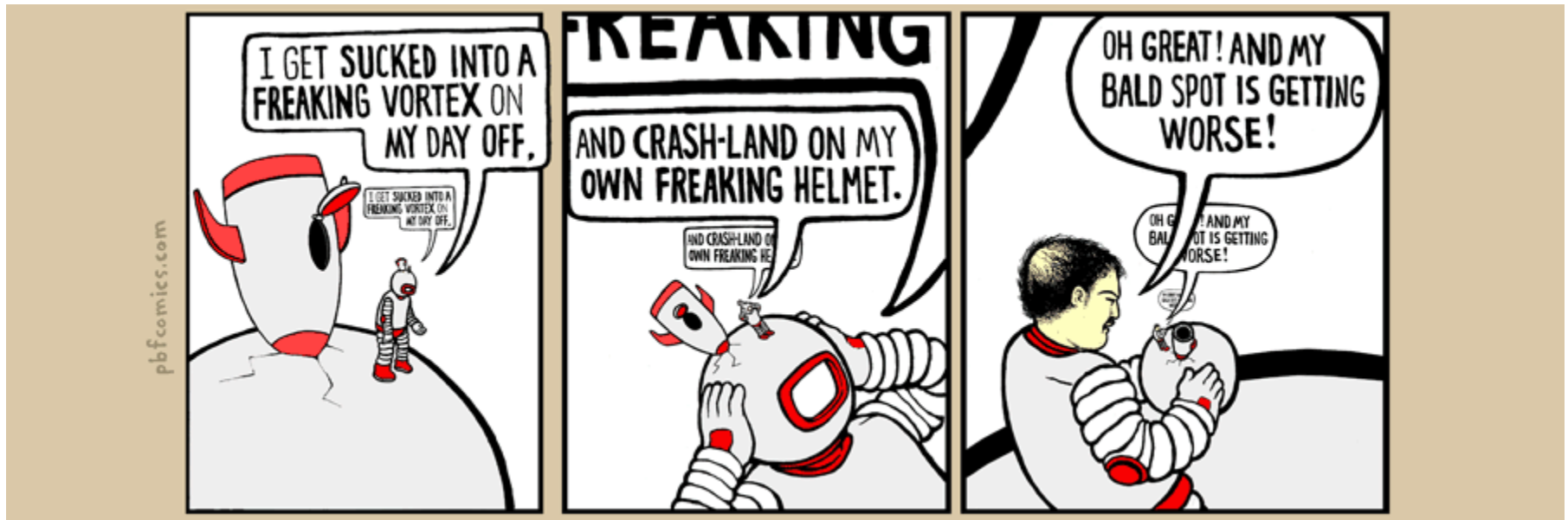
- Een static variable kan enkel een value zijn, **nooit een expression** (vb. **voorbeeld-functions-static**)

Functions

- Scope
 - Opdracht: opdracht-functies-gevorderd

Functions

- Recursive functions



Functions

- Recursive function
 - recursief = zichzelf aanschrijven.
 - Een functie kan zichzelf dus aanschrijven.
 - OPGEPAST: dit kan een infinite loop veroorzaken en dus een crash van het script.
 - (vb. [voorbeeld-functions-recursie-simpel](#))
 - (vb. [voorbeeld-functions-recursie-return](#))
 - (vb. [voorbeeld-functions-recursie-uitgebreid](#))
 - Opdracht: [opdracht-recursive](#)

\$_GET

- \$_GET variable
 - Manier om informatie naar de server te sturen
 - Zichtbaar voor iedereen
 - Zichtbaar in de url
 - Voordeel: kan makkelijk gedeeld worden

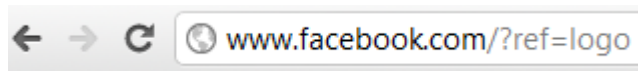


- Informatiegrootte is gelimiteerd (max. 2000 karakters)

\$_GET

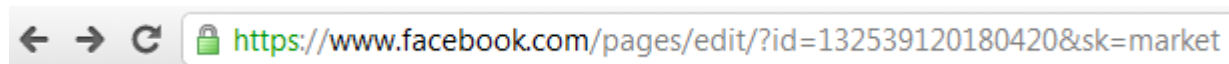
- \$_GET variable
 - Wordt gebruikt voor het '**tonen van informatie**'
 - Syntax:
 - Eén variable:

`http://www.url.be/contact.html?variablename=value`



- Meerdere variables:

`http://www.url.be/?variablename=value&variablename02=value02`



\$_GET

- Hoe gebruiken?

- form.html

```
<form action="validate.php" method="get">  
    <input type="text" name="email">  
</form>
```

- validate.php

```
$_GET['email'];    => is dus een ARRAY!
```

- (vb. [voorbeeld-get-basis](#))

- Opdracht: [opdracht-get](#)

\$_POST

- \$_POST variable
 - Manier om informatie naar de server te sturen
 - Enkel zichtbaar voor de server
 - Informatiegrootte is niet gelimiteerd
 - Wordt gebruikt voor het **aanpassen** van informatie en voor het doorsturen van **gevoelige informatie** (usernames/passwords)

\$_POST

- Hoe gebruiken?

- form.html

```
<form action="validate.php" method="post">  
    <input type="password" name="password">  
</form>
```

- validate.php

```
$_POST['password'];    => is dus een ARRAY!
```

- (vb. **voorbeeld-post-basis**)

Controle \$_GET & \$_POST

- Wanneer \$_GET of \$_POST worden aangeroepen zonder dat er iets 'gesubmit' is
=> error-message!

Notice: Undefined index: ... in ... on line ...

Controle \$_GET & \$_POST

- Controleren of een key in een array bestaat:

```
if ( isset( $_POST[ 'key' ] ) )  
{  
    uit te voeren code;  
}
```

(vb. [voorbeeld-get-post-key-controle](#))

- Opdracht: [opdracht-post](#)

Controle \$_GET & \$_POST

- Controles (kunnen gebruikt worden als condition)
 - `array_key_exists()`
 - `in_array()`
 - `empty()`
 - `isset()`
 - `$variableName`
 - `! $variableName` (\Leftrightarrow `$variable`)
 - ... (php.net)

Herhalingsoefening

opdracht-herhalingsopdracht-01