

PHP

Key concepts

PHP, server, client, requests, variabelen, var_dump, gettype, cast-operaties, operaties =, &=, == vs ===, +, -, *, %, ., if, for, while, array, key-value pairs, functions, type hinting.

Alternatieve bronnen

<http://php.net/docs.php>

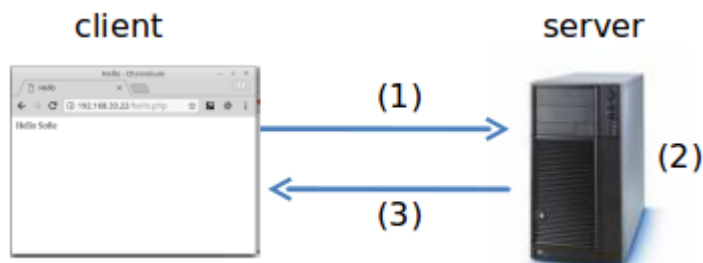
[https://www.codecademy.com/courses/web-beginner-en-StaFQ/0/1?](https://www.codecademy.com/courses/web-beginner-en-StaFQ/0/1?curriculum_id=5124ef4c78d510dd89003eb8)

[curriculum_id=5124ef4c78d510dd89003eb8](https://www.w3schools.com/php/default.asp)

<https://www.w3schools.com/php/default.asp>

1. Inleiding

PHP is een server-sided scripttaal die gebruikt wordt om dynamische webpagina's te maken. Een eenvoudige voorstelling van de uitvoering van een PHP-programma wordt getoond in onderstaande figuur. Via de browser (client) wordt een request naar de server gestuurd (1). De PHP interpreter op de server voert het programma uit (2) en het resultaat wordt als response teruggestuurd naar de browser (3).



De broncode die op de server staat wordt hieronder getoond. Deze code staat in de map /var/www/html/ .

hello.php

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Hello</title>
</head>
<body>
<?php
$name = 'Sofie';
print( "Hello $name\n" );
?>
</body>
</html>
```

Buiten HTML-code bevat het bestand ook PHP-code. Deze code staat in een PHP-tag `<?php ... ?>`. Bij het uitvoeren van de code vervangt de interpreter elke PHP-tag door de output die in de tag gegenereerd wordt.

De HTML-code ontvangen door de browser wordt hieronder getoond (ctrl-u in Chrome).



Via Chrome developer tools (ctrl-shift-i) kan de communicatie in detail bekeken worden:

The screenshot shows the Chrome Developer Tools interface with the 'Network' tab selected. The browser window displays 'Hello Sofie'. The address bar shows '192.168.33.22/hello.php'. The Network tab shows a list of requests, with 'hello.php' selected. The 'Headers' sub-tab is active, displaying the following information:

- General**
 - Request URL: http://192.168.33.22/hello.php
 - Request Method: GET
 - Status Code: 200 OK
 - Remote Address: 192.168.33.22:80
 - Referrer Policy: no-referrer-when-downgrade
- Response Headers** (view parsed)
 - HTTP/1.1 200 OK
 - Date: Tue, 16 Jan 2018 12:52:13 GMT
 - Server: Apache/2.4.18 (Ubuntu)
 - Vary: Accept-Encoding
 - Content-Encoding: gzip
 - Content-Length: 125
 - Keep-Alive: timeout=5, max=100
 - Connection: Keep-Alive
 - Content-Type: text/html; charset=UTF-8
- Request Headers** (view parsed)
 - GET /hello.php HTTP/1.1
 - Host: 192.168.33.22
 - Connection: keep-alive
 - Cache-Control: max-age=0
 - Upgrade-Insecure-Requests: 1
 - User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.84 Safari/537.36
 - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
 - Accept-Encoding: gzip, deflate
 - Accept-Language: en-US,en;q=0.9

At the bottom, it shows '1 requests | 389 B transferred ...'.

De client stuurt een request (GET /hello.php HTTP/1.1), de server stuurt een response (HTTP/1.1 200 OK) met daarin de HTML-code die getoond moet worden in de browser.

2. Variabelen

In PHP wordt een variabele aangeduid door een dollarteken (\$) gevolgd door een aantal geldige symbolen. Geldige symbolen omvatten letters, hoofdletters, cijfers en de underscore (_). Een variabele mag echter niet met een cijfer beginnen. Variabelen zijn 'case-sensitive' (hoofdlettergevoelig): \$getal en \$GETAL zijn twee verschillende variabelen.

In dit hoofdstuk worden variabelen van het type boolean, integer, double, string en NULL besproken. Booleans bevatten logische waarden, integers gehele getallen, doubles kommagetallen en strings tekstwaarden (het datatype char bestaat niet in PHP). NULL is de null-referentie. Meer informatie over de datatypes is terug te vinden in de documentatie¹. (Let bij het datatype string vooral op het verschil tussen single en double quotes. Bij single quotes wordt de inhoud letterlijk overgenomen, bij double quotes worden variabelen en worden geëscape'te symbolen zoals \n geëxpandeerd).

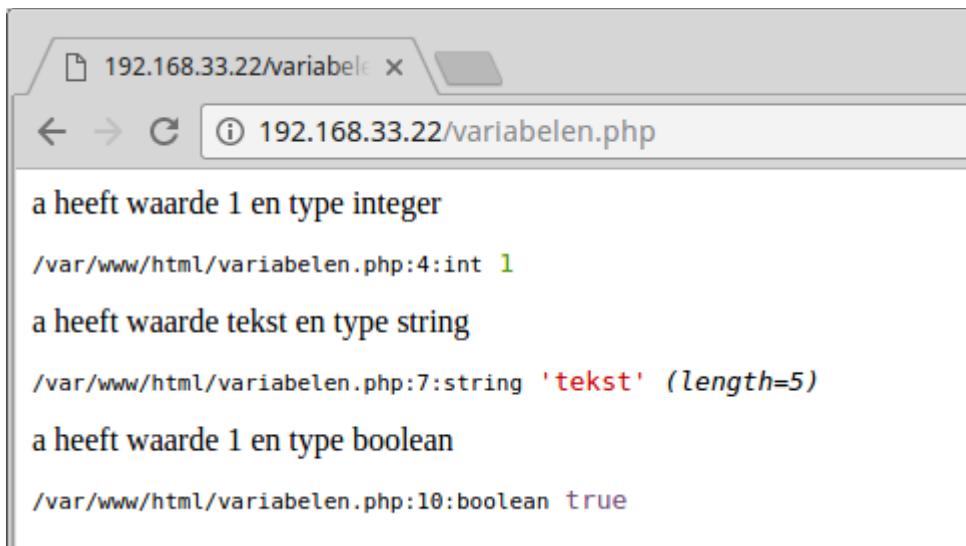
PHP is een flexibele programmeertaal: variabelen worden niet gedeclareerd en elke variabele kan waarden van eender welk type bevatten. Het type van de variabele wordt bepaald door de context. In het onderstaande voorbeeld (variabelen.php) worden waarden van verschillende datatypes toegekend aan de variabele \$a. Het type wordt opgevraagd via de functie gettype en via de functie var_dump² wordt de volledige informatie van de variabele afgedrukt.

variabelen.php

```
<?php
$a = 1;
print ('a heeft waarde ' . $a . ' en type ' . gettype($a) .
'<br/>');
var_dump($a);
$a = "tekst";
print ('a heeft waarde ' . $a . " en type ' . gettype($a) .
'<br/>');
var_dump($a);
$a = TRUE;
print ('a heeft waarde ' . $a . ' en type ' . gettype($a) .
'<br/>');
var_dump($a);
```

¹<http://php.net/manual/en/language.types.intro.php>

²<http://php.net/manual/en/function.var-dump.php>



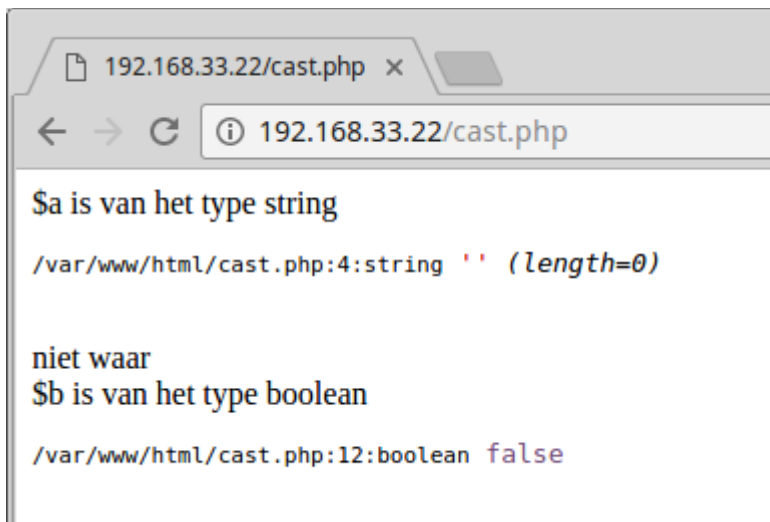
```
192.168.33.22/variabelen.php
a heeft waarde 1 en type integer
/var/www/html/variabelen.php:4:int 1
a heeft waarde tekst en type string
/var/www/html/variabelen.php:7:string 'tekst' (length=5)
a heeft waarde 1 en type boolean
/var/www/html/variabelen.php:10:boolean true
```

3. Cast-operaties

Het is ook mogelijk omzettingen tussen verschillende types te maken via cast-operaties. In het onderstaande voorbeeld (cast.php) krijgt de variabele \$a een lege string als waarde. De inhoud van de variabele \$a wordt vervolgens via een cast-operatie omgezet naar het type boolean en het resultaat wordt in de variabele \$b geplaatst.

cast.php

```
<?php
$a = "";
print ("\$a is van het type " . gettype($a));
var_dump($a);
$b = (boolean)$a;
if ($b) {
    print ("</br>waar <br/>");
} else {
    print ("<br>niet waar <br/>");
}
print ("\$b is van het type " . gettype($b));
var_dump($b);
```



```
192.168.33.22/cast.php x
192.168.33.22/cast.php
$a is van het type string
/var/www/html/cast.php:4:string '' (length=0)

niet waar
$b is van het type boolean
/var/www/html/cast.php:12:boolean false
```

In het voorbeeld wordt aangetoond dat een lege string wordt omgezet naar de boolean false. De cast-regels naar datatypes zijn terug te vinden in de documentatie³.

Kort samengevat:

De cast naar boolean gebeurt via de operaties (boolean) of (bool). De volgende waarden worden naar false gecast:

- de integer 0
- de double 0.0
- een lege string en de string met waarde "0"
- Null

Alle andere waarden van het type integer, double of string worden naar true gecast.

De cast naar integer gebeurt via de operaties (integer) of (int). De volgende regels gelden voor deze operatie:

- true wordt naar 1 gecast, false naar 0
- een double wordt naar beneden afgerond (12.34 wordt 12)
- een string die niet begint met een cijfer wordt naar 0 omgezet ("test 123" wordt 0). een string die wel begint met een aantal cijfers wordt omgezet naar het overeenkomende getal. Spaties in het begin van de string worden hierbij genegeerd. Ook worden symbolen na de reeks cijfers niet in rekening gebracht. (" 123aad" wordt 123).
- Null wordt 0

De cast naar float verloopt analoog aan de cast naar integer.

De omzetting naar string gebeurt via de cast-operatie (string). Van numerieke waarden wordt eenvoudig een string-representatie gegeven. De boolean true wordt omgezet naar "1" en de boolean false naar een lege string. Ook Null wordt omgezet naar de lege string.

³<http://php.net/manual/en/language.types.boolean.php#language.types.boolean.casting>
<http://php.net/manual/en/language.types.integer.php#language.types.integer.casting>
<http://php.net/manual/en/language.types.float.php#language.types.float.casting>
<http://php.net/manual/en/language.types.type-juggling.php>

4. Operaties

Via de toekenningsoperatie ('=') krijgt een variabele een waarde en wordt ook het type van de variabele bepaald. Via '&' kan er assign by reference een toekenning gedaan worden. In onderstaand voorbeeld verwijzen de variabelen \$a en \$c naar dezelfde locatie in het geheugen, wijzigingen in \$c worden ook gemerkt in \$a;

variabelen.php

```
<?php
$a = 1;
$b = $a;
$c = & $a;
$b = 2;
$c = 3;
print ($a . ' ' . $b . ' ' . $c . "\n");
```

```
jan@laptop-jan ~/Desktop/code/test $ php variabelen.php
3 2 3
```

Meer informatie is te vinden in de documentatie ⁴.

De belangrijkste wiskundige operaties zijn +, -, *, /, %. Deze operaties verwachten altijd twee numerische argumenten en het resultaat is steeds numerisch.

Als bij de bewerkingen +, - en * een van de argumenten een float is (of een string die kan omgezet worden naar een double) dan worden beide beschouwd als float en is het resultaat een float. Anders worden beide geëvalueerd als integer en is het resultaat een integer.

variabelen.php

```
<?php
var_dump(1+"1.22aaaa");
```

```
jan@laptop-jan ~/Desktop/code/test $ php variabelen.php
/home/jan/Desktop/code/test/variabelen.php:2:
double(2.22)
```

Bij een deling worden beide argumenten als float beschouwd en is het resultaat een float.

Bij de modulo-bewerking worden de argumenten steeds als integer beschouwd en is het resultaat een integer.

⁴<http://php.net/manual/en/language.operators.assignment.php>
extra: <https://derickrethans.nl/talks/phparch-php-variables-article.pdf>

De belangrijkste tekst-operatie is de concatenatie (.) die twee strings aan elkaar verbindt. De argumenten links en rechts van deze operatie worden beide als string beschouwd en indien nodig naar string gecast. In onderstaand voorbeeld wordt true gecast naar “1” en false gecast naar “”.

variabelen.php

```
<?php
var_dump('-' . true . '-' . false . '-');
```

```
jan@laptop-jan ~/Desktop/code/test $ php variabelen.php
/home/jan/Desktop/code/test/variabelen.php:2:
string(4) "-1--"
```

Met behulp van vergelijkingsoperaties worden twee waarden vergeleken. Het resultaat van zo een operatie is steeds een boolean: true of false naargelang de waarden al dan niet voldoen aan de voorwaarde. De belangrijkste vergelijkingsoperaties worden gegeven in de volgende tabel:

==	gelijk in waarde
===	gelijk in waarde en van hetzelfde datatype
!=	verschillend in waarde
!==	verschillend in waarde of datatype
<	kleiner dan
>	groter dan
<=	kleiner dan of gelijk aan
>=	groter dan of gelijk aan

Er wordt een onderscheid gemaakt tussen de vergelijkingen == en ===. De operatie == controleert of twee waarden gelijk zijn, === controleert of twee waarden gelijk zijn en hetzelfde type hebben.

variabelen.php

```
<?php
var_dump(1=='1');
var_dump(1==='1');
```

```
jan@laptop-jan ~/Desktop/code/test $ php variabelen.php
/home/jan/Desktop/code/test/variabelen.php:2:
bool(true)
/home/jan/Desktop/code/test/variabelen.php:3:
bool(false)
```

De logische operaties (&, &&, |, ||) zijn dezelfde als in Java⁵.

5. Controle structuren

Op de details na zijn de controlestructuren (if, for, while, do while) vergelijkbaar met die in Java⁶. Een voorbeeld wordt hieronder gegeven.

controle.php

```
<?php
$a=0;
if ( $a< 1 ) {
    print("$a is kleiner dan 1\n");
} else {
    print("$a is groter dan 1\n");
}

while( $a< 10 ) {
    print($a);
    $a++;
}
print("\n");

for ( $i = 0 ; $i < 10 ; $i++ ){
    print($i);
}
print("\n");
```

```
jan@laptop-jan ~/Desktop/code/test $ php controle.php
0 is kleiner dan 1
0123456789
0123456789
```

⁵<http://php.net/manual/en/language.operators.logical.php>

⁶<http://php.net/manual/en/language.control-structures.php>

6. Arrays

In PHP is een array een datatype waarin een aantal waarden (values) bewaard kunnen worden. Bij elke waarde in de array is er een sleutel (key) voorzien waarmee de waarde opgevraagd kan worden. (In veel programmeertalen wordt deze datastructuur een 'map' genoemd.)

Een array kan aangemaakt worden via de functie array:

```
$a = array(element_1, element_2, ... ,element_N);
```

waarbij

```
element_i = value_i  
element_i = key_i => value_i
```

In deze definitie wordt de rij \$a bestaande uit N elementen aangemaakt. Per element wordt ofwel enkel de waarde (value i) gespecificeerd ofwel wordt de combinatie van sleutel (key i) en waarde (value i) opgegeven.

De waarden (values) in een array hoeven niet allemaal van hetzelfde type te zijn: in eenzelfde rij kunnen waarden van verschillende types bewaard worden. De sleutel (key) moet van het datatype integer of string zijn. Als er geen key gesp wordt bij het aanroepen van de functie array dan wordt de eerstvolgende beschikbare integer genomen als key.

arrays.php

```
<?php  
$a = array(1, 2, 3, 4);  
$b = array(1, 2.1, true, "ja");  
$c = array(1 => 12, "Juist" => true);  
$d = array (2 => 1, 10);  
  
echo '<table border ="1">';  
echo '<tr><td>$a = array(1,2,3,4);</td><td> ';  
var_dump($a);  
echo '</td></tr>';  
echo '<tr><td>$b = array(1,2.1,true,"ja");</td><td> ';  
var_dump($b);  
echo '</td></tr>';  
echo '<tr><td>$c = array(1=> 12, "Juist" => true);</td><td> ';  
var_dump($c);  
echo '</td></tr>';  
echo '<tr><td>$d = array (2 => 1, 10);</td><td> ';  
var_dump($d);  
echo '</td></tr>';  
echo '</table>';
```

\$a = array(1,2,3,4);	/var/www/html/arrays.php:9: array (size=4) 0 => int 1 1 => int 2 2 => int 3 3 => int 4
\$b = array(1,2.1,true,"ja");	/var/www/html/arrays.php:12: array (size=4) 0 => int 1 1 => float 2.1 2 => boolean true 3 => string 'ja' (length=2)
\$c = array(1=> 12, "Juist" => true);	/var/www/html/arrays.php:15: array (size=2) 1 => int 12 'Juist' => boolean true
\$d = array (2 => 1, 10);	/var/www/html/arrays.php:18: array (size=2) 2 => int 1 3 => int 10

Bij de creatie van de rijen \$a en \$b werden enkel waarden (values) ingegeven.

De key die bij elke value hoort wordt automatisch gegenereerd, deze keys gaan in beide gevallen van 0 tot 3. Voor de rij \$c werd er bij elk element zowel een key als een waarde gespecificeerd. De key 1 bij de waarde 12 en de key Juist bij de waarde true.

Bij de definitie van de rij \$d werd zowel een key,value-paar (key 2 en value1) als een value (10) ingegeven. De value 10 krijgt key 3 omdat 3 de eerstvolgende beschikbare integer key is (na de key 2).

Na de creatie van een array kunnen waarden (values) opgehaald en gewijzigd worden aan de hand van hun sleutels (keys). Een waarde uit de rij wordt geselecteerd via vierkante haken:

```
$rij[ key_i ]
```

selecteert de value die overeenkomt met key_i.

Een voorbeeld van het gebruik van keys wordt hieronder gegeven:

arrays.php

```
<?php
$a = array(1, 2, 3, 4);
echo "$a[0]\n";
$a[0] = "ja";
var_dump($a);
echo "\n";
$b = array("een" => 1, "twee" => 3 );
echo $b["een"] . "\n";
echo "$b[een]\n";
$b["twee"] = 2;
var_dump($b);
echo "\n";
```

```
jan@laptop-jan ~/Desktop/code/test $ php arrays.php
1
array(4) {
  [0] => fine, but in rare cases it can
  string(2) "ja"
  [1] =>
  int(2) sure the guest additions within t
  [2] =>
  int(3) of VirtualBox you have installed
  [3] =>
  int(4)
}
1
1
/home/jan/Desktop/code/test/arrays.php:11:
array(2) {
  'een' => vagrants/vagrant_new
  int(1) p/werk/vagrants/vagrant_new/shared
  'twee' => grant provision` or use the `--
  int(2)
}. Provisioners marked to run always will
```

In dit voorbeeld wordt de waarde met key 0 van de rij \$a eerst afgedrukt en daarna vervangen door de string "ja";
Van de rij \$b wordt de waarde met key 'een' afgedrukt . De waarde met key 'twee' wordt vervolgens vervangen door 2.

Zoals getoond in het volgend voorbeeld kunnen na de creatie van de rij nog steeds elementen bij de rij toegevoegd worden.

arrays.php

```
<?php
$a = array( 1, 2, 3, 4 );
$a[7] = 11;
var_dump( $a );
echo "\n";

$b = array( "een" => 1, "twee" => 3 );
$b["drie"] = 3;
var_dump( $b );
echo "\n";
```

```
jan@laptop-jan ~/Desktop/code/test $ php arrays.php
/home/jan/Desktop/code/test/arrays.php:4:
array(5) { ..
  [0] => int(1)
  [1] => int(2)
  [2] => int(3)
  [3] => int(4)
  [7] => int(11)
}
5.1.22
/home/jan/Desktop/code/test/arrays.php:9:
array(3) {
  'een' => int(1)
  'twee' => int(3)
  'drie' => int(3)
}
```

Bij de rij \$a wordt de value 11 met als key 7 toegevoegd, bij de rij \$b wordt de value 3 met als key "drie" geplaatst.

Ook via lege vierkante haken kan een waarde toegevoegd worden:

```
$rij [ ] = value_i;
```

De waarde value_i wordt in de rij geplaatst met als key de eerstvolgende vrije index.

In het volgend voorbeeld wordt de waarde nieuw toegevoegd met 1 als key (de key 0 was al ingenomen door de waarde 1).

arrays.php

```
<?php
$c = array( 1, "a" =>2 );
$c[] ="nieuw";
var_dump( $c );
```

```
jan@laptop-jan ~/Desktop/code/test $ php arrays.php
/home/jan/Desktop/code/test/arrays.php:4:
array(3) {
  [0] =>
  int(1)
  [1] =>
  string(5) "nieuw"
```

Elementen kunnen ook verwijderd worden uit de rij via de functie unset. Zie onderstaand voorbeeld:

arrays.php

```
<?php
$d = array( 1, "a" =>2, "b" );
unset( $d["a"] );
var_dump( $d );
```

```
jan@laptop-jan ~/Desktop/code/test $ php arrays.php
/home/jan/Desktop/code/test/arrays.php:4:
array(2) {
  [0] =>
  int(1)
  [1] =>
  string(1) "b"
```

Enkele nuttige functies die kunnen toegepast worden op een array worden hieronder opgesomd:

<code>count(\$a)</code>	geef het aantal waarden in de rij \$a
<code>array_keys(\$a)</code>	geef een rij met alle keys van \$a
<code>array_keys(\$a, val)</code>	geef een rij met de keys die overeenkomen met de waarde val (zoekfunctie)
<code>array_values(\$a)</code>	geef een rij met alle values van \$a
<code>sort(\$a)</code>	sorteer de rij \$a
<code>shuffle(\$a)</code>	schud de rij \$a
<code>min(\$a)</code>	minimum van de rij \$a
<code>max(\$a)</code>	maximum van de rij \$a

7. Foreach

foreach een handige structuur die gebruikt kan worden om door de waarden (en de sleutels) van een rij te gaan. Via een eerste versie van de foreach-lus wordt elke waarde van de rij \$rij in de variabele \$v gekopieerd.

```
foreach( $rij as $v ) {  
  
}
```

foreach.php

```
<?php  
$a = array(1 => "ma", 2 => "di", 3 => "wo", 4 => "ma");  
foreach ($a as $v) {  
    echo "value: $v \n";  
}
```

```
ajan@laptop-jan ~/Desktop/code/test $ php foreach.php  
value: ma  
value: di  
value: wo  
value: ma
```

Via een tweede vorm van de foreach-lus is zowel de waarde (\$v) als de sleutel (\$k) van elk element uit de rij beschikbaar in de sequentie.

```
foreach ($rij as $k => $v) {  
  
}
```

foreach.php

```
<?php  
$a = array(1 => "ma", 2 => "di", 3 => "wo", 4 => "ma");  
foreach ($a as $k=>$v) {  
    echo "key: $k, value: $v \n";  
}
```

```
jan@laptop-jan ~/Desktop/code/test $ php foreach.php  
key: 1, value: ma  
key: 2, value: di  
key: 3, value: wo  
key: 4, value: ma
```


8. Functies

Een functie bevat een blok code die uitgevoerd wordt wanneer de functie aangeroepen wordt. Bij het aanroepen kunnen argumenten meegegeven worden en er kan ook een waarde teruggeven worden via het keyword `return`⁷.

functions.php

```
<?php
function som ( $getal1, $getal2 ) {
    return $getal1 + $getal2;
}

function printResultaat ( $resultaat ) {
    print ( $resultaat );
}

$resultaat = som ( 1 , 2 );
printResultaat ( $resultaat );
```

Elke naam kan slechts één keer gebruikt worden om een functie te definiëren. Er zijn twee manieren hoe men deze beperking kan vermijden: de eerste methode bestaat erin een default argument voor de functie som te voorzien. Een default waarde is een argument waarvan een waarde wordt voorzien. In het onderstaand voorbeeld heeft het argument `$getal3` default waarde 0. De functie kan worden aangeroepen met twee argumenten (`$getal3` is dan gelijk aan 0), maar hij kan ook aangeroepen met drie argumenten.

functions.php

```
<?php
function som ( $getal1, $getal2, $getal3 = 0 ) {
    return $getal1 + $getal2 + $getal3;
}

$resultaat = som ( 1 , 2 );
print ( $resultaat );
$resultaat = som ( 1 , 2 , 3 );
print ( $resultaat );
```

⁷<http://php.net/manual/en/functions.user-defined.php>

Een tweede oplossing bestaat er in geen argument in te geven bij de declaratie en gebruik te maken van de functie `func_get_args`⁸. Er hoeft nu geen argument vermeld te worden bij de definitie van de functie. De functie kan sowieso met eender welk aantal argumenten aangeroepen worden. Deze argumenten worden opgevraagd en in een array geplaatst via de functie `func_get_args`.

functions.php

```
<?php
function som ( ) {
    $som=0;
    $argumenten=func_get_args();
    foreach ( $argumenten as $argument ) {
        $som += $argument;
    }
    return $som;
}

$resultaat = som ( 1 , 2 );
print ( $resultaat );
$resultaat = som ( 1 , 2 , 3, 7, 9);
print ( $resultaat );
```

Net zoals in Java kan ook de 'splat' - operator gebruikt worden

functions.php

```
<?php
function som ( ...$getallen ) {
    $som=0;
    foreach ( $getallen as $getal ) {
        $som += $getal;
    }
    return $som;
}

$resultaat = som ( 1 , 2 );
print ( $resultaat );
$resultaat = som ( 1 , 2 , 3, 7, 9);
print ( $resultaat );
```

Argumenten kunnen ook pass by reference doorgegeven worden, zie documentatie ⁹.

⁸<http://php.net/manual/en/function.func-get-args.php>

⁹<http://php.net/manual/en/functions.arguments.php>

Sinds PHP5 kunnen argumenten getypehint worden: arrays en klassen kunnen als datatype bij argumenten vermeld worden:

functions.php

```
<?php
function printDatum ( DateTime $date ) {
    print ($date->format('Y-m-d H:i:s')) ;
}

function printRij ( array $rij){
    foreach ($rij as $element){
        print($element."\n");
    }
}

$date=new DateTime();
printDatum($date);

$rij=[1,2,3];
printRij($rij);
```

Vanaf PHP7 geldt dit ook voor de datatypes string, int, float, boolean. Ook kan de return-type vermeld worden bij de functie

functions.php

```
<?php
function som (int $getal1, int $getal2 ) : int {
    return $getal1 + $getal2;
}

$resultaat = som ( 1 , 2 );
print ( $resultaat );
```

8. Voorgedefinieerde functies

In deze sectie worden enkele voorgedefinieerde functies besproken.

String functies

<code>strlen(\$s)</code>	lengte van de string \$s
<code>strpos (\$s, \$z)</code>	eerste positie van een zoekstring \$z in de string \$s false als \$z niet gevonden wordt
<code>substr(\$s, \$i)</code>	substring van de string \$s, beginnende vanaf positie \$i
<code>strtolower (\$s)</code>	geeft de string \$s omgezet naar kleine letters terug
<code>strtoupper (\$s)</code>	geef de string \$s omgezet naar hoofdletters
<code>trim(\$s)</code>	geef de string \$s zonder spaties voor en achteraan

Functies voor variabelen

<code>define ("PI", 3.1415)</code>	definieer een constante
<code>unset (\$a)</code>	verwijder de variabele \$a
<code>isset(\$a)</code>	bestaat de variabele \$a?
<code>is_bool(\$a)</code>	is de variabele \$a een boolean?
<code>is_double(\$a)</code>	double?
<code>is_string(\$a)</code>	string?
...	
<code>get_type(\$a)</code>	geef het type van het variabele \$a

Include en require

Via de functies `include` en `require` wordt een bestand geopend, de PHP-code in dit bestand wordt uitgevoerd en het resultaat wordt geplaatst op de positie waar de functie aangeroepen werd. Het verschil tussen `include` en `require` ligt in hoe fouten behandeld worden. Indien het te openen bestand niet bestaat geeft `include` een warning terwijl `require` een fatale error geeft. `include_once` en `require_once` zijn vergelijkbaar met `include` en `require`. Enkel wordt er nu voor gezorgd dat elk bestand maar 1 keer ingevoegd wordt. Via deze functies kunnen programma's opgesplitst worden in meerdere bestanden.

math.php

```
<?php
function som (int $getal1,int $getal2 ):int {
    return $getal1 + $getal2;
}
```

som.php

```
<?php
require_once ( 'math.php' );
$resultaat = som ( 1, 2 );
print ( $resultaat ) ;
```

Header

Via de functie header kan de response header van de server gewijzigd worden. Belangrijk is dat het aanroepen van de functie gebeurt voor output verstuurd wordt. De functie header wordt daarom meestal bovenaan in een PHP-bestand aangeroepen.

Header kan gebruikt worden om een header redirect te maken. Onderstaande regel zorgt ervoor dat de browser doorverwezen wordt naar een andere pagina. De opdracht exit() zorgt ervoor dat de rest van het huidige script niet uitgevoerd wordt.

header.php

```
<?php
header( "Location: http://www.demorgen.be/" );
exit();
```

Via de functie header kan ook het MIME-type van de verstuurde inhoud bepaald worden. In onderstaand voorbeeld wordt een bestand als PDF verstuurd.

header.php

```
<?php
$file = 'a.pdf';
header('Content-type: application/pdf');
header('Content-Disposition: attachment; filename="'.
$file .'"');
readfile($file);
```