PHP
Het maken van webapplicaties

Huub de Beer

Eindhoven, 4 juni 2011, versie 0.2

Inhoudsopgave

0 Inleiding

Een geschiedenis van het WWW • het statische internet • het dynamische internet • CGI en het begin van PHP • de handleiding online

0.0 Een korte geschiedenis van het web en PHP

Tim Berners Lee ontwikkelde in de jaren '80 van de vorige eeuw het *World Wide Web* (WWW) om onderzoeksinformatie op het CERN toegankelijk te maken voor alle medewerkers van het Zwitserse onderzoeksinstituut. Het WWW bestond uit vier basiscomponenten: het Hypertext Transfer Protocol (**HTTP**), een aantal afspraken hoe webpagina's van de ene computer naar de andere computer te transporteren; Hypertext Markup Language (**HTML**), een taal om webpagina's in op te stellen; een **webbrowser** om webpagina's te bekijken en een **webserver** om webpagina's op te vragen.

In 1991 maakte Berners Lee zijn werk publiek via een newsgroup op het Internet. Verschillende onderzoeksinstituten en universiteiten pikten het idee op en breidden het WWW verder uit. Het web bestond voornamelijk uit statische informatie: tekst, afbeeldingen en hyperlinks tussen verschillende pagina's. Een auteur schreef een webpagina en plaatste die op een webserver. Een gebruiker kon vervolgens die pagina opvragen bij de server die dan het bestand met de webpagina las en de inhoud naar de gebruiker stuurde. We noemen dit het **statische internet**.

Al snel liepen de webpioniers tegen de beperkingen van het statische internet aan: een hele hoop informatie paste niet in het statische stramien. Zou het niet handig zijn als de gegevens uit een database via het WWW opgevraagd kunnen worden? Een gebruiker geeft aan welke gegevens uit de database hij wil hebben en de webserver haalt die op uit de database en genereert dan een webpagina met die gegevens. Hierop ontwikkelden een aantal vroege webgebruikers in 1993 de **Common Gateway Interface** (CGI) waarmee via het web programma's op de server kunnen worden uitgevoerd. Zo'n programma kan dan een verbinding maken met een database en de opgevraagde informatie omzetten in een webpagina.

Op het moment dat een programma op verzoek van een gebruiker een webpagina genereert, spreken we van een dynamische webpagina. Het dynamische internet is dan het statische internet gecombineerd met dergelijke dynamische webpagina's. Alhoewel CGI programma's in elke programmeertaal geschreven kunnen worden, waren juist dynamische programmeertalen zoals *perl* en *tcl* populair. Was snelheid van groot belang, dan kozen ontwikkelaars ook vaak voor de systeemprogrammeertaal C.

Bij het schrijven van CGI programma's moet een programmeur keer op keer dezelfde functionaliteit inbouwen; veel CGI programma's lijken wat betreft hun structuur op elkaar. Het duurde dan ook niet lang voordat verschillende projecten ontstonden om die standaard functionaliteit aan programmeurs van CGI programma's aan te bieden. Een van deze projecten was *PHP*, dat in 1994/1995 niet veel anders was dan een aantal standaard programma's op de homepage van de bedenker, Rasmus Lerdorf. Dit PHP-systeem evolueerde halverwege de jaren negentig tot de *PHP Hypertext Preprocessor* (PHP). Ondertussen zijn er verschillende nieuwe versies van PHP verschenen, maar de basis is nog steeds hetzelfde: eenvoudig toegang tot heel veel functionaliteit.

0.1 PHP

In de handleiding van PHP op http://www.php.net/manual.en vind je meer informatie over de vele mogelijkheden van PHP. Tot die mogelijkheden behoren toegang tot verschillende databases, het genereren van afbeeldingen

1

Webapplicaties I

First Contact

Webapplicaties met HTML en PHP • een eerste PHP script • over POST en GET • van sleutels en associaties

Op het internet kom je allerlei webapplicaties tegen: een roosterprogramma van een school, een reserveringsservice van de bibliotheek, een webwinkel, enzovoorts. In een webapplicatie verloopt de interactie met de gebruiker via HTML formulieren. Met behulp van zo'n formulier vraagt de applicatie aan de gebruiker om bepaalde gegevens in te vullen. De gebruiker klikt na het invullen op de *submit*-knop en zijn webbrowser verstuurt de ingevoerde gegevens naar de webserver waarop de webapplicatie draait.

In Listing 1.0 vindt je de HTML code van een eenvoudig formulier (`NAW.html`) waarin de gebruiker zijn NAW gegevens invult. NAW staat voor Naam, Adres en Woonplaats. De gegevens die de gebruiker in dit formulier invult, worden doorgestuurd naar het PHP script `verwerkNAW.php`. Dit geef je in het HTML formulier aan met behulp van de parameter action="verwerkNAW.php" van het element <form> (zie regel 7).

```
1  <html>
2    <head>
3      <title>Een eenvoudig NAW formulier</title>
4    </head>
5    <body>
6      <h2>Vul je NAW gegevens in:</h2>
7      <form action='verwerkNAW.php' method='POST>
8        Naam:       <input type='text' name='naam'> <br>
9        Adres:      <input type='text' name='adres'> <br>
10       Woonplaats: <input type='text' name='woonplaats'> <br>
11       <input type='submit' value='insturen'>
12     </form>
13   </body>
14 </html>
```

Listing 1.0: NAW.html: Een eenvoudig NAW formulier.

De webapplicatie gaat vervolgens die gegevens verwerken met behulp van het PHP script `verwerkNAW.php` (zie Listing 1.1). De communicatie tussen het HTML formulier en het PHP script verloopt via het *associatieve array* $_POST of $_GET. In welke array de ingevoerde gegevens te vinden zijn, hangt af van de *methode* van verzenden. Die methode geef je aan in het HTML formulier met de parameter method="POST" of method="GET" van het element <form> (zie regel 7).

Meestal wordt voor de methode *POST* gekozen omdat die methode van verzenden veiliger is dan *GET*. De methode GET plakt namelijk de te verzenden gegevens achteraan de URL van het PHP script dat de gegevens verwerkt. De URL ziet er dan ongeveer zo uit:
`verwerkNAW.php?naam=Johan&adres=Kerkstraat+2&woonplaats=Eindhoven`. De POST methode verstuurt de gegevens "ongezien" naar de webserver. Natuurlijk kunnen ook de met POST verstuurde gegevens afgeluisterd worden.

```
1  <?php
2  // verwerkNAW.php: gemaakt door Huub de Beer, 20−04−2009, versie 0
3  $naam = $_POST['naam'];
4  $adres = $_POST['adres'];
5  $woonplaats = $_POST['woonplaats'];
6  ?>
7  <html>
8    <head>
9      <title>Je NAW gegevens</title>
10   </head>
11   <body>
12     <h2>Welkom <?php echo "$naam"; ?></h2>
13     <p>
14       Je woont op dit moment op de <?php echo "$adres"; ?> te <?php echo
15       "$woonplaats"; ?>.
16     </p>
17   </body>
18 </html>
```

Listing 1.1: verwerkNAW.php: PHP verwerkt de invoer van NAW.html

In het voorbeeld worden de ingevulde gegevens met de methode POST verzonden en het PHP script kan die gegevens vinden in het associatieve array `$_POST`. Als *Martha Jozefs*, woonachtig op de *Kerkstraat 23a* te *Eindhoven* het formulier invult en op de "insturen"-knop klikt, ziet het `$_POST` array er als volgt uit:

sleutel	**waarde**
'naam'	'Martha Jozefs'
'adres'	'Kerkstraat 23a'
'woonplaats'	'Eindhoven'

De sleutels van dit array zijn de namen van de controls in het HTML formulier. Die namen zijn met behulp van de parameter `name='naam'`, `name='adres'` en `name='woonplaats'` van de elementen `<input type='text'>` ingesteld. De door de gebruiker ingevulde waarden kun je met behulp van deze sleutels opvragen uit het array `$_POST`: je plaatst tussen rechte haken de sleutel achter de variabelenaam van het array.

In Listing 1.1 worden voor de naam, adres en woonplaats nieuwe variabelen geïntroduceerd en van de door de gebruiker ingevoerde waarde voorzien (regels 3–5). Het is gebruikelijk om voor alle waarden uit `$_POST` die je in je PHP script gebruikt nieuwe variabelen te introduceren. Dat maakt je code een stuk duidelijker. Een variabele herken je aan het `$`-teken: elke variabelenaam begin met een `$`-teken. Na het introduceren van de nieuwe variabelen geeft `verwerkNAW.php` die gegevens vervolgens in een HTML pagina weer.

1.0 Praktische opgaven

1.0.0 Typ `NAW.html` (Listing 1.0, pagina 4) en `verwerkNAW.php` (Listing 1.1, pagina 5) over en voer ze uit.

1.0.1 Herschrijf beide bestanden uit de vorige opgave zodanig dat de gegevens die de gebruiker invoert met behulp van de methode GET worden verzonden en niet meer met de methode POST.

1.0.2 Maak een webapplicatie waarmee de gebruiker zijn leeftijd kan uitrekenen (analoog aan het voorbeeld in dit hoofdstuk). De gebruiker vult zijn naam en geboortedatum in en de webapplicatie berekent vervolgens zijn leeftijd:

a. Maak een HTML formulier met twee invoervelden, een voor de naam van de gebruiker en een voor zijn geboortejaar. Sluit het formulier af met een submit-knop. Noem dit formulier: `geboortejaar.html`. De methode van verzenden is POST.

b. Schrijf een PHP script dat de door de gebruiker ingevoerde gegevens verwerkt. Introduceer nieuwe variabelen $naam en $geboortejaar. Noem dit script: `leeftijd.php`.
Introduceer vervolgens een nieuwe variabele $leeftijd waarin je met een eenvoudige expressie de leeftijd van de gebruiker berekent: zijn leeftijd is het huidige jaar minus zijn geboortejaar.
Tot slot informeer je de gebruiker met een volledige HTML pagina over zijn leeftijd. Spreek de gebruiker persoonlijk aan: je kent nu immers zijn naam (opgeslagen in de variabele $naam).

c. ⋆ Werkt je PHP script volgend jaar ook nog? En over tien jaar? Zo nee, pas je script dan aan. Gebruik de PHP functie **date**() om altijd het huidige jaar te gebruiken. Zoek de functie **date**() eerst op in de PHP handleiding: http://www.php.net/manual/en/function.date.php.

⋆ **1.0.3** Kleine kinderen mogen graag aftellen tot een grote gebeurtenis zoals een verjaardag. Maak een "Nog N Nachtjes slapen" webapplicatie.

a. Vraag de gebruiker om zijn naam, de gebeurtenis en de datum in te vullen met behulp van een HTML formulier.

b. Na verwerking van deze gegevens krijgt de gebruiker een resultaatpagina te zien waarop is aangegeven hoeveel nachtjes hij nog moet slapen voordat de "grote gebeurtenis" plaatsvindt.
Maak gebruik van de PHP functies **strtotime**(*"datum"*) en **time**(). Beide functies geven een zogenaamde Unix *timestamp* terug, het aantal seconden sinds 1 januari 1970. Trek beide timestamps van elkaar af en je krijgt het verschil tussen nu en de ingevoerde datum in seconden. Je kunt dan zelf het aantal dagen uitrekenen.

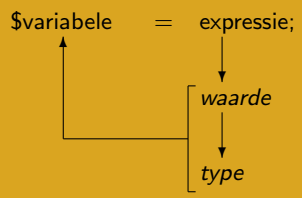
2 De toekenning: over variabelen en expressies

Variabelen • toekenning van waarde • en automatisch een type • eenvoudige expressies: literal, variabele en functie • complexe expressies met haakjes en operatoren

De basiseenheid van informatie in een programma is de variabele. Variabelen gebruik je om resultaten van (tussen)berekeningen op te slaan of om langere stukken tekst op te bouwen. Met behulp van variabelen krijg je toegang tot informatie van de gebruiker of uit een database. De variabele is het werkpaard van een programmeertaal.

Variabelen in PHP herken je aan het $-teken; elke variabelenaam begint ermee. Variabelen zijn bergruimten in het geheugen van de computer met een unieke naam. In de bergruimte kan een waarde worden opgeslagen. Hoe die variabelen precies in het geheugen worden bewaard, is verder niet van belang. Daar zorgt PHP voor. Het is jouw taak om PHP te vertellen welke variabelen aangemaakt moeten worden, PHP doet de rest.

toekenning



Een variabele geef je een nieuwe waarde door de **toekenning** met behulp van het =-teken. Links van het =-teken staat altijd een variabele. Rechts van het =-teken schrijf je een **expressie**. Zo'n expressie heeft een waarde en die waarde behoort tot een van de vier basistypen van PHP: integer, float, string of Boolean, of een samengesteld type zoals een (associatief) array. Door de toekenning krijgt een variabele zowel de waarde van de expressie als het type van die waarde toegekend.

$variabele *wordt* expressie. De *waarde* van de expressie wordt toegekend aan de variabele. De variabele krijgt dan automatisch het *type* van de waarde van de expressie. PHP kent vier basistypen: *integer*, *float*, *string* en *Boolean*:

basistype	**omschrijving**	**voorbeelden**
integer	gehele getallen	−34, 456, 0
float	komma getallen	3.14, −0.00034. 0.23453E−12
string	stukjes tekst	*"Hallo lezer"*. *"A"*, *'$var is een variabele '*
Boolean	waarheidswaarden	**true** en **false**

Om het type van een variabele $var te weten te komen kun je de PHP functie **gettype**($var) gebruiken. Om type en waarde van een variabele af te drukken, bijvoorbeeld bij het *debuggen* van je PHP script, gebruik je de functie **var_dump**($var);.

Het is belangrijk in te zien dat het =-teken hier **geen** gelijkheid betekent zoals in de wiskunde gebruikelijk is. Het =-teken heeft een richting: het transporteert de waarde (en type) van de expressie aan de rechterkant naar de variabele aan de linkerkant. We zeggen dan ook niet "$getal is 15" maar "$getal *wordt* 15".

Het is verstandig variabelen een *betekenisvolle* naam te geven. Je bent volledig vrij in het kiezen van namen zolang ze maar met een letter of een underscore (_) beginnen. Niets weerhoudt je ervan om je variabelen $var1, $var2, enzovoorts te noemen. Het is echter onduidelijk en bij meer dan twee variabelen is het een bron van onnodige fouten.

De meest eenvoudige expressies zijn letterlijke waarden zoals 23, **false**, *"Hallo lezer"* of 2.65234. Dergelijke letterlijke waarden noemen we **literals**. Enkele voorbeelden:

```
1 $punt = 6.7; // $punt krijgt de waarde 6.7 met type float
2 $naam = "Jan Wesselink"; // $naam krijgt de waarde "Jan Wesselink"
3     //met type string
4  $aantal_ll  = 26; // $aantal_ll krijgt de waarde 26 met type integer
5 $grootste = true; // $grootste krijgt de waarde true met type Boolean
```

Naast literals zijn er nog twee andere soorten eenvoudige expressies: variabelen en functies (zie hoofdstuk 8):

```
1 $nieuw_punt = $punt; // $nieuw_punt krijgt de waarde die de variabele
2     // $punt heeft met het daarbij horende type ( float , zie vorig voorbeeld)
3  $trilling  = sin( 0.3245 ); // $trilling krijgt de waarde de sinus van 0.3245
4     // berekend door de functie sin() en heeft als type float ;
5 $vandaag = date( "d-m-Y" ); // $vandaag krijgt de waarde de datum van
6     // vandaag als een string met het formaat ''12-9-2009'' zoals de
7     // functie date() dat ''berekent''
```

De waarde van de variabelen aan de rechterkant van het =-teken wordt gebruikt in de expressie. Met behulp van deze drie eenvoudige expressies is het mogelijk om complexe expressies op te bouwen met behulp van ronde haakjes en *operatoren*:

```
$gemiddelde = ( $punt1 + $punt2 + $punt3 ) / 3; // $gemiddelde krijgt
    // als waarde het gemiddelde van de drie punten en heeft type
    // float of integer, al naargelang de som van de drie punten
    // een geheeld getal is en deelbaar door 3
$voldoende = ($gemiddelde >= 5.5); // Het gemiddelde is een voldoende
    // als het groter of gelijk is aan 5.5
$uitvoer = $naam . " heeft een " . $gemiddelde . " gehaald voor Frans.";
    // $uitvoer wordt een string opgebouwd uit de naam,
    // de string " heeft een", het gemiddelde en de string
    // " gehaald voor Frans."
$opp_cirkel = 3.14 * 2 * $r; // $opp_cirkel wordt 2*pi*r met type float
```

De meeste operatoren ken je al, bijvoorbeeld uit de wiskunde, en spreken voor zich. Andere operatoren zijn nieuw, zoals de .-operator die twee strings aan elkaar koppelt. Als een expressie bestaat uit elementen met verschillende typen, dan *converteert* PHP alle elementen eerst tot één type en berekent dan de expressie. Dat kan soms tot vreemde resultaten leiden, dus pas op!

2.0 Gemengde opgaven

2.0.0 Welke waarde en type krijgt de variabele $var toegekend in onderstaande gevallen? Maak onderstaande opgaven zonder PHP te gebruiken.

a. $var = 12 * 12;
b. $var = − 9.0;
c. $var = "Welk type is dit?";
d. $var = (sin(0.67) >= 0.01);
e. $var = 12 / 3;
f. $var = 13 / 3;
g. $var = "Dertien" . " + 12";
h. $var = 0.12 + 12 + 0.88;
i. $var = 12 + 120 + 88;
j. $var = 12.0 + 120 + 88;

2.0.1 Controleer je antwoorden van de vorige opgave met behulp van een PHP script en de PHP functie var_dump($var). Zijn er verrassende antwoorden?

3 Basistypen en hun operatoren

Getallen: integer en float • stukjes tekst: string • waarheidswaarden: Boolean

3.0 Getallen: *integer* en *float*

PHP kent twee verschillende typen getallen: integer en float. Integer getallen zijn gehele getallen. Float getallen zijn komma getallen en getallen in de wetenschappelijke notatie zoals 0.2344E−8. Beide typen kennen dezelfde standaard rekenkundige operaties zoals vermenigvuldigen, optellen, aftrekken en delen.

Het type integer kent daarnaast ook nog de operatie *modulo*. De expressie 20 modulo 3 heeft waarde 2 omdat $\frac{20}{3} = (6 * 3) + 2$. We zeggen dan dat 2 de rest is na deling van $\frac{20}{3}$. De modulo operator levert dus de rest na deling op.

rekenkundige operatoren

De typen *float* en *integer* kennen de volgende operatoren:

operator	omschrijving	voorbeeld
−	min-teken	$negatief = − 23;
+	optellen	$som = 12 + 4.5 + 7;
−	aftrekken	$som = 12 − 4.5 − (2 − 45);
∗	vermenigvuldigen	$product = 2 ∗ 4 ∗5.5;
/	delen	$deling = 20 / 3;
%	modulo (integer)	$rest = 20 % 3;

Deze operatoren kunnen gecombineerd worden met het =-teken tot de volgende *toekenninsoperatoren*: +=, −=, ∗=, /= en %=.

Voor het verhogen of verlagen van de waarde van een variabele met 1 bestaan de operatoren ++ en −−.

PHP kent ook een groot aantal standaard wiskundige functies:
http://www.php.net/manual/en/ref.math.php.

Een expressie opgebouwd uit waarden van verschillende typen, dus zowel integer waarden als float waarden, heeft **altijd** het type float. Alleen als een expressie bestaat uit alleen maar integer waarden, dan is het type van die expressie ook integer. De uitzondering op deze regel is de deling: als twee gehele getallen op elkaar worden gedeeld die niet deelbaar zijn dan is het resultaat toch een float.

In programma's kom je vaak toekenningen tegen van de vorm: $var = $var + $waarde;. Met zo'n toekenning update je een variabele: je geeft het een nieuwe waarde gebaseerd op de oude waarde. In dit geval is het een optelling, maar het had net zo goed een vermenigvuldiging, deling of aftrekking kunnen zijn. Om het leven van de programmeur gemakkelijker te maken, kent PHP zogenaamde *toekenningsoperatoren*. Een toekenningsoperator bestaat uit het =-teken voorafgegaan door de gewenste operator zoals +, −, ∗, / of %:

```
1 $teller = 0;
2 // Op drie manieren een erbij optellen:
3 $teller = $teller + 1;
4 $teller += 1;
5 $teller ++;
6 // Op twee manieren een er van af trekken:
7 $teller --;
8 $teller -= 1;
```

Naast de toekenningsoperatoren kent PHP ook de operatoren ++, ophogen met 1, en −−, verlagen met 1. Daarmee kun je de toekenningen $var = $var + 1; of $var = $var − 1; nog korter opschrijven.

3.1 Stukjes tekst: *string*

Het basistype voor tekst is de *string*. Er zijn twee manieren om string literals te maken: enkele en dubbele aanhalingstekens (*quotes*).[1] Het verschil tussen enkele en dubbele aanhalingstekens zit in *substitutie* van speciale tekens en variabelen.

String literals omsloten door enkele aanhalingstekens kennen geen speciale tekens en vervangen variabelen *niet* door de waarde van die variabelen. String literals met dubbele aanhalingstekens substitueren variabelen *wel* door de waarde van die variabelen en kennen daarnaast ook een aantal speciale tekens zoals de nieuwe regel ("\n") of de tab ("\t"). Enkele voorbeelden:

```
1 $punt = 6.7;
2 $str = 'Je hebt een $punt gehaald\ ngefeliciteerd !';
3 echo $str; // -> Je hebt een $punt gehaald\ ngefeliciteerd !
4 $str = "Je hebt een $punt gehaald";
5 echo $str; // -> Je hebt een 6.7 gehaald
6            // -> gefeliciteerd !
```

[1] Naast enkele en dubbele aanhalingstekens bestaan nog twee manieren om string literals te maken: heredoc en newdoc. Deze worden hier niet besproken, voor meer informatie zie: http://www.php.net/manual/en/language.types.string.php.

string operatoren

Het type *string* kent de *concatenatie* operator (.) om twee strings aan elkaar te knopen. Waarden van andere typen worden voordat ze aan elkaar geknoopt worden eerst geconverteerd naar het type string. Getallen worden omgezet naar een tekstuele representatie van dat getal, dus 23.45 wordt omgezet naar *"23.45"*. De waarheidswaarde **true** wordt omgezet naar *"1"* en **false** naar de *lege string* (*""*).

De concatenatie operator kan met het =-teken gecombineerd worden tot de toekenningsoperator .=

PHP kent een groot aantal string bewerkingsfuncties:
http://www.php.net/manual/en/ref.strings.php.

```
1  $brief = "";
2  $naam = 'Jan';
3  $afzender = "Marie";
4  $aantal = 5;
5
6  $brief = "Beste " . $naam . ', ';
7  $brief .= "\n\n";
8  $brief .= "bij dezen zend ik jou $aantal appels.";
9  $brief .= "\n" . ' ... ' . "\n";
10 $brief = $brief . "\n met vriendelijke groet,\n\n$afzender";
```

3.2 Waarheidwaarden: *Boolean*

Het type *Boolean* kent precies twee waarden: **true**, "waar", en **false**, "onwaar". Boolean expressies, behalve de literals **true** en **false** en functies die waarden van het type Boolean teruggeven, komen bijna nooit in toekenningen voor. Toch zijn Boolean expressies zeer belangrijk, ze zijn een essentieel onderdeel van de selectie en de repetitie (zie Hoofdstuk 5 en 9).

Boolean operatoren

Het type *Boolean* kent de volgende operatoren:

operator	omschrijving	voorbeeld
!	*not* (niet)	$onwaar = ! **true**;
and	*and* (én)	$beide_gevallen = $geval1 and $geval2;
or	*or* (óf)	$een_of_beide = $geval1 or $geval2;

De werking van deze operatoren kan worden weergegeven met onderstaande waarheidstabel:

$a	$b	!$a	$a and $b	$a or $b
false	**false**	**true**	**false**	**false**
false	**true**	**true**	**false**	**true**
true	**false**	**false**	**false**	**true**
true	**true**	**false**	**true**	**true**

Met behulp van deze operatoren en haakjes is het mogelijk complexe Boolean expressies op te stellen. Boolean expressies worden pas interessant als die ex-

pressies uitspraken doen over andere variabelen en expressies. Dergelijke expressies noemen we *condities*. Om uitspraken te doen over andere variabelen en expressies gebruik je *vergelijkings operatoren*.

vergelijkings operatoren

operator	omschrijving	voorbeeld
<	kleiner dan	$onwaar = (5 < 4);
<=	kleiner dan of gelijk aan	$waar = (5.5 <= 6.4);
==	gelijk aan	$onwaar = ("hallo" == "");
>=	groter dan of gelijk aan	$waar = (2.3 >= 2.3);
>	groter dan	$onwaar = (2.3 > 22);
!=	ongelijk aan	$waar = (3 != 2.5);

Met behulp van de vergelijkings operatoren <, <=, ==, >=, > en != kun je expressies met elkaar vergelijken. Daarmee kun je onderzoeken of een variabele groter of gelijk is aan 5 of dat een variabele niet gelijk is aan de lege string (""). Let op, bij vergelijking van expressies met verschillende typen wordt een van de expressies geconverteerd naar het type van de andere expressie. Dat kan soms tot vreemde en onverwachte resultaten leiden.

Nu begrijp je waarschijnlijk ook de verwarring die het =-teken oproept. In PHP geven we gelijkheid namelijk aan met == en niet met =. In de wiskunde en het dagelijks leven ben je echter gewend om het =-teken wel te gebruiken om gelijkheid aan te geven. Om de verwarring te verminderen spreken we het =-teken uit als "wordt" en de ==-operator als "is gelijk aan".

Enkele voorbeelden van Boolean expressies:

```
$punt = 5.6;
$voldoende = ( $punt >= 5.5 ); // $voldoende == true
$onvoldoende = ( $punt < 5.5 ); // $onvoldoende == false

// Het verschil tussen and en or is groot
 $altijd_fout  = $voldoende and $onvoldoende;
 $altijd_waar  = ( $punt >= 5.5 ) or ( $punt < 5.5 );

$lange_expr = (
     (( true or false ) and true and ( false or true )) or (false or true)
  ); // Welke waarde heeft deze expressie ?

// Ook complexe expressies kun je vergelijken
$groter = ( $punt > ( ( 12 * 5.1 ) / 13.5 ) );
$mix = ( $naam != "" ) and ( $punt <= 8 and $punt >= 7 );
```

3.3 Gemengde opgaven

3.3.0 Geef zonder PHP te gebruiken aan wat de waarde is van variabele $var in onderstaande opgaven. De waarde verandert bij elke opgave, gebruik de nieuwe waarde van $var om de waarde in de volgende opgave te bepalen.

a. $var = −3;
b. $var *= 2;
c. $var += $var − ($var * 2);
d. $var −= −5;
e. $var += $var++;
f. $var++;
g. $var /= 3;
h. $var %= 2;
i. $var −= 3;

Kijk vervolgens je antwoorden na met behulp van PHP.

3.3.1 Geef zonder PHP te gebruiken aan wat de waarde is van de Boolean variabele $var in onderstaande opgaven. Vertaal vervolgens de Boolean expressie in gewoon Nederlands: welke eigenschap of conditie wordt uitgedrukt?

Gegeven is de volgende PHP code:

```
$getal = 320;
$punt = 7.5;
$kleur = 'rood';
```

a. $var = ($getal % 10 == 0);
b. $var = ($punt >= 5.5 and $punt < 8);
c. $var = ($getal % $getal == 1);
d. $var = ($kleur == "zwart"or $kleur == "wit");
e. $var = (($punt + 1) <= 10);
f. $var = ($kleur != "zwart");

3.3.2 Schrijf PHP code voor onderstaande opgaven zonder PHP te gebruiken. Gebruik voor onbekende waarden een variabele: kies een geschikte naam en introduceer de variabele door toekenning van een literal.

De eerste opgave wordt voorgedaan:

```
<?php
$klant = "Jan Cornelisz";
$aantal_boeken = 13;
 $stukprijs = 34.45;
 $bestelling = "$klant heeft $aantal_boeken boeken besteld";
 $bestelling .= " voor een totaalprijs van ";
 $bestelling .= ($aantal_boeken * $stukprijs ) . " Euro.";
?>
```

a. Ken aan de variabele $bestelling een zin met de naam van de klant, het aantal bestelde PHP boeken en de prijs toe. De prijs wordt berekend door het aantal bestelde boeken en de stukprijs.
b. Bepaal in de Boolean variabele $is_even of een getal even is.
c. Bereken in een variabele het gemiddelde punt van een klas van 5 leerlingen. Introduceer voor elke leerling een nieuwe variabele.
d. Deze opgave bouwt voort op de vorige opgave. De klas is geslaagd als elke leerlingen een voldoende heeft gehaald. Bepaal in de Boolean variabele $geslaagd of deze klas geslaagd is.
e. Bereken in vier verschillende variabelen de omtrek ($2\pi r$) en oppervlakte (πr^2) van een cirkel en de oppervlakte ($4\pi r^2$) en inhoud ($\frac{4}{3}\pi r^3$) van een bol. Gebruik de PHP functie **pi**() om de waarde van π te verkrijgen.
f. Maak in de string variabele $tabel de volgende tabel na:

	-1	1
cos	$cos(-1)$	$cos(1)$
sin	$sin(-1)$	$sin(1)$
tan	$tan(-1)$	$tan(1)$

Gebruik de PHP functies **sin**(), **cos**() en **tan**() om sinus, cosinus en tangens voor de waarden -1 en 1 uit te rekenen. Je trekt een horizontale streep door het $-$-teken een aantal keer te herhalen. Gebruik "\n" om een nieuwe regel te beginnen.

g. $\star$ Bepaal in de Boolean variabele $schrikkeljaar of een jaar een schrikkeljaar is.

4

Webapplicaties II

Meer controls

HTML formulieren bestaan niet alleen uit tekstvakjes en submit knoppen. Er zijn meer controls die je allemaal in je webapplicatie kunt gebruiken: grote tekstvakken, keuzerondjes, aankruisvakjes en keuzelijsten. Aankruisvakjes worden later besproken.

```
<html>
  <head>
    <title>Enqu&ecirc;te computergebruik</title>
  </head>
  <body>
    <h1>Enqu&ecirc;te computergebruik</h1>
    <form action="verwerk_enquete.php" method="POST">
    <ol>
      <li>Naam: <input type="text" name="naam"></li>
      <li>Leeftijd: <input type="text" name="leeftijd"></li>
      <li>
        Sekse:
        <input type="radio" name="sekse" value="man">man
        <input type="radio" name="sekse" value="vrouw">vrouw
      </li>
      <li>
        Welke processorarchitectuur gebruik je het liefst ?<br>
        <input type="radio" name="arch" value="x86" checked>x86
        <input type="radio" name="arch" value="arm">arm
        <input type="radio" name="arch" value="powerpc">powerpc
        <input type="radio" name="arch" value="sparc">sparc
      </li>
      <li>
        Welk besturingssysteem gebruik je het meest?
        <select name="os">
          <option value="linux">Linux</option>
          <option value="mac">Mac OSX</option>
          <option value="react">React OS</option>
          <option value="solaris">Solaris</option>
          <option value="windows">Windows</option>
        </select>
      </li>
      <li>
        Waarom gebruik je een computer?<br>
        <textarea name="waarom" rows="5" cols="50">
        </textarea>
```

```
37        </li>
38      </ol>
39      <input type="submit" value="insturen">
40    </body>
41  </html>
```

Listing 4.0: enquete.html: een korte enquete over computergebruik

In dit hoofdstuk gaan we een webapplicatie bouwen om een korte enquête over computergebruik af te nemen. In Listing 4.0 zie je de HTML code van de enquête. Alhoewel de HTML code van de verschillende controls anders is dan die van een tekstvakje, gaat de koppeling met PHP precies hetzelfde: je geeft elke control een eigen unieke naam.

Een groep keuzerondjes vormt dan samen één control. Alle keuzerondjes in dezelfde groep krijgen dan ook dezelfde naam als waarde voor de parameter name van het element <input>.

Naast een naam krijgen sommige elementen, zoals <option> en <input type="radio">, ook een waarde voor de parameter value mee (regels 11–32). De browser zendt de waarde van het geselecteerde keuzerondje of item uit de keuzelijst naar de webserver als waarde van de hele control.

```
1  <?php
2  // verwerkenquete.php
3  // auteur: Huub de Beer
4  // versie: 0.2, 2009
5  $naam = $_POST['naam'];
6   $leeftijd  = $_POST['leeftijd']  ];
7  $sekse = $_POST['sekse'];
8  $arch = $_POST['arch'];
9  $os = $_POST['os'];
10 $waarom = $_POST['waarom'];
11 ?>
12 <html>
13   <head>
14     <title>Verwerkte enqu&ecirc;te computergebruik</title>
15   </head>
16   <body>
17     <h1>Verwerkte enqu&ecirc;te computergebruik</h1>
18     <p>
19 <?php
20 echo "$naam ($leeftijd jaar oud) gebruikt $os op $arch.";
21 if ( $sekse == "man" ) {
22   echo " Hij gebruikt zijn computer omdat:";
23 } elseif ( $sekse == "vrouw" ) {
24   echo " Zij gebruikt haar computer omdat:";
25 } else {
26   // De gebruiker heeft geen sekse ingevuld:
27   echo " Het gebruikt zijn computer omdat:";
28 };
29 ?>
30     </p>
31     <blockquote>
32       <?php echo "$waarom\n"; ?>
33     </blockquote>
```

```
34   </body>
35 </html>
```

Listing 4.1: verwerk_enquete.php: de korte enquête over computergebruik verwerkt

De enquête wordt verwerkt door het PHP script `verwerk_enquete.php` (Listing 4.1). Zoals gewoonlijk introduceer je nieuwe variabelen voor de door de gebruiker ingevulde waarden. Je vindt die waarden in het associatieve array $_POST.

Op regels 17–24 zie je een if-statement dat aan de hand van de door de gebruiker ingevulde sekse een zin afdrukt. Met behulp van een Booleaanse expressie wordt getest of de ingevulde sekse, *"man"*, *"vrouw"* of iets anders is. In elk geval wordt dan andere PHP code uitgevoerd: PHP kan keuzen maken. Je leert in een van de volgende hoofdstukken meer over het if-statement.

4.0 Praktische opgaven

4.0.0 Maak een bestelpagina voor boekwinkel *Scherper dan het zwaard*. Die bestelpagina, `boekbestelpagina.html`, is een aantrekkelijk opgemaakt HTML formulier. Op de bestelpagina vraag je de klant om zijn of haar NAW gegevens (tekstvakjes).

Vervolgens kan de klant uit een lijst met boeken een keuze maken (keuzelijst). Elk boek in die lijst heeft een andere prijs. Deze prijs is de waarde van de parameter value van de <**option**>-elementen in de keuzelijst. Daarna heeft de gebruiker de mogelijkheid om meerdere exemplaren te bestellen, standaard is 1 ingevuld (tekstvakje).

De klant kiest uit een viertal alternatieven om de bestelling te betalen (keuzerondjes), credit card, iDEAL, PayPal of overschrijving. Tot slot kan de gebruiker nog enige opmerkingen achterlaten (groot tekstvak).

Schrijf vervolgens het PHP script dat bestellingen die via deze bestelpagina binnenkomen, verwerkt. Dit PHP script, `boekfactuur.php`, geeft een overzichtelijke factuur van de bestelling op het scherm weer. De factuur bestaat uit de NAW gegevens, het bestelde boek met prijs en aantal, de standaard verzendkosten van 3.45 €, het totaalbedrag en de opmerkingen van de klant.

4.0.1 (zie ook de opgaven van het vorige hoofdstuk)

Maak een webapplicatie om de omtrek, oppervlakte en inhoud van cirkel en bol uit te rekenen. Maak een HTML formulier, kies zelf een geschikte naam, waarin de gebruiker uit een keuzelijst kan kiezen tussen de eigenschappen omtrek cirkel, oppervlakte cirkel, oppervlakte bol en inhoud bol. Laat de gebruiker ook een straal invoeren.

Het script dit HTML formulier verwerkt, rekent vervolgens de gevraagde eigenschap uit. Gebruik een if-statement om onderscheid te maken tussen de verschillende eigenschappen. Gebruik de PHP functie **pi**() om de waarde van π te verkrijgen.

5 PHP bewandelt verschillende wegen

5.0 Het if-statement

Met behulp van het if-statement is het mogelijk om bepaalde PHP code wel of niet uit te voeren al naargelang de waarde van een Booleaanse expressie. Zo'n Booleaanse expressie noemen we een *conditie*. Het if-statement maakt het mogelijk om een script te schrijven dat om kan gaan met een onbekende waarde, zoals de invoer van een gebruiker. Je kunt anticiperen op mogelijke waarden en voor elk geval, voor elke conditie, geschikte PHP code schrijven. Een voorbeeld:

```
<?php
$punt = $_POST[ 'punt' ];
if ( $punt < 5.5 ) {
  // $punt is een onvoldoende
  echo "FAIL!!!";
} else {
  // $punt is een voldoende
  echo "WIN!!!";
};
?>
```

De gebruiker voert op een formulier een punt in. PHP reageert op die invoer: voert de gebruiker een getal kleiner dan 5.5 in, dan geeft PHP "FAIL!!!" weer. In het andere geval, dus als het punt ten minste een 5.5 is, voert PHP andere code uit: `echo "WIN!!!;"`.

Dit is een vrij zinloze applicatie, maar het geeft wel heel goed weer wat het if-statement doet:

if-statement

```
if ( conditie ) {
  // conditie geldt
  // Blok A
} else {
  // conditie geldt *niet*
  // Blok B
};
```

Een *conditie* is een Booleaanse expressie. Heeft die expressie de waarde **true**, dan zeggen we dat die conditie *geldt*. In het andere geval geldt de conditie *niet*.
Als de conditie geldt, voert PHP de code in Blok A uit. Geldt de conditie niet dan voert PHP de code in Blok B uit. Blok A en Blok B worden dus *conditioneel* uitgevoerd.

Een *blok* is een groep PHP statements omsloten door accolades ({ en }). Je begint een blok met een { op dezelfde regel als de bijbehorende controlestructuur. Je eindigt het blok met een } op hetzelfde *inspringniveau* als de bijbehorende controlestructuur. De PHP statements in het blok *springen* een niveau extra *in*.

Het vorige voorbeeld kan uitgebouwd worden tot een applicatie om beoordelingen in cijfers om te zetten naar beoordelingen in tekst. Er wordt dan niet alleen onderscheid gemaakt tussen onvoldoende en voldoende; een punt wordt vertaald naar een item uit de lijst: slecht, onvoldoende, voldoende, goed en uitmuntend:

```
1  <?php
2  $punt = $_POST[ 'punt' ];
3
4  // vertaal $punt naar een beoordeling:
5  if ( $punt <= 4 ) {
6    echo "slecht";
7  } elseif ( $punt < 5.5 ) {
8    // $punt tussen 4 en 5.5
9    echo "onvoldoende";
10 } elseif ( $punt < 7 ) {
11   // $punt tussen 5.5 en 7
12   echo "voldoende";
13 } elseif ( $punt < 9 ) {
14   // $punt tussen 7 en 9
15   echo "goed";
16 } else {
17   // $punt tussen 9 en 10:
18   echo "uitmuntend";
19 };
20 ?>
```

Met **elseif** kun je meerdere condities in een if-statement testen. PHP voert alleen het blok met PHP statements uit dat bij de *eerst* geldende conditie hoort. Als geen enkele conditie geldt dan voert PHP het blok na **else** uit.

Niets weerhoudt je ervan om complexe booleaanse expressies als conditie te gebruiken. Meestal word code er niet duidelijker op. Probeer lange condities

te vermijden. Een voorbeeld met complexe condities is het beoordelen of een leerling over gaat, doubleert of de laatste rapportvergadering de leerling gaat bespreken:

```
<?php
$aantal_onvoldoendes = $_POST[ 'onvoldoendes' ];
$gemiddelde  = $_POST[ 'gemiddelde' ];
if ( $aantal_onvoldoendes < 2 and $gemiddelde >= 5.5 ) {
  echo "Je bent over";
} elseif ( ($aantal_onvoldoendes >= 2 and $gemiddelde >= 5.5) or
           ($gemiddelde >= 5 and $aantal_onvoldoendes <= 2 ) ) {
  echo "Je bent een bespreekgeval";
} else {
  echo "Je blijft zitten";
};
?>
```

5.1 Het switch-statement

Naast het if-statement kent PHP een ander conditioneel statement: het switch-statement.

switch-statement

```
switch ( $var ) {
  case waarde_1:
    echo '$var is gelijk aan waarde_1';
    break;
  case waarde_2:
    echo '$var is gelijk aan waarde_2';
    break;
  // ...
  case waarde_N:
    echo '$var is gelijk aan waarde_N';
  default:
    // In alle andere gevallen
    echo 'geen enkele waarde is gelijk aan de variabele ';
};
```

Het switch-statement test of de waarde van $var overeenkomt met een van de waarden na **case** en voert de daarbij horende code uit. Als geen enkel geval overeenkomt, voert PHP de code bij **default** uit.

Schrijf aan het einde van elke **case** het sleutelwoord **break** om te voorkomen dat PHP ook de andere gevallen gaat bekijken.

Het switch-statement is handig voor variabelen die enkel waarden uit een kleine lijst met waarden kunnen aannemen. Bijvoorbeeld een variabele die de dag van de week bevat: een van zeven mogelijkheden. Of een variabele die de maand in het jaar bevat, een van twaalf mogelijkheden. Of een variabele die

een kaart in een reeks kaarten (1 op 13) of het aantal ogen van een dobbelsteen (1 op 6):

```php
<?php

// genereer een willekeurig getal tussen 1 en 6 met de functie rand
// met andere woorden: gooi een dobbelsteen:
$ogen = rand( 1, 6 );

switch ( $ogen ) {
  case 1:
    echo "Je hebt een <b>Ace</b> gegooid.";
    break;
  case 2:
    echo "Je hebt een <b>Deuce</b> gegooid.";
    break;
  case 3:
    echo "Je hebt een <b>Trey</b> gegooid.";
    break;
  case 4:
    echo "Je hebt een <b>Cater</b> gegooid.";
    break;
  case 5:
    echo "Je hebt een <b>Cinque</b> gegooid.";
    break;
  case 6:
    echo "Je hebt een <b>Sice</b> gegooid.";
    break;
  default:
    // zal nooit voorkomen
};
?>
```

Alle waarden na het case keyword in een switch-statement hebben of het type integer of het type string. Voor meer voorbeelden van het switch-statement, zie de PHP handleiding: http://www.php.net/manual/en/control-structures.switch.php.

5.2 Gemengde opgaven

5.2.0 Maak onderstaande opgaven zonder PHP te gebruiken, maar zodanig dat je het PHP script wel eenvoudig kunt testen. Indien mogelijk, controleer je werk.

a. Schrijf PHP code waarin je de dag van de week opvraagt met behulp van de functie **date**('l') (de kleine letter L is het argument van de functie **date**()). Vertaal vervolgens die dag van de week van het Engels naar het Nederlands. Print tot slot de frase "Het is vandaag zondag" waarbij "zondag" natuurlijk de door PHP vertaalde dag is.
   Let op, namen van de dagen beginnen in het Engels met een hoofdletter en in het Nederlands niet.

b. Gegeven is het volgende stukje HTML code:

```
1  <!-- ... -->
2  <form action="..." method="POST">
3    <input type="text" name="linkeroperand">
4    <select name="operator">
5      <option value="plus">+</option>
6      <option value="min">-</option>
7      <option value="maal">*</option>
8      <option value="deel">/</option>
9    </select>
10   <input type="text" name="rechteroperand">
11   <input type="submit" value="bereken">
12 </form>
13 <!-- ... -->
```

Schrijf PHP code om bovenstaand formulier te verwerken. Dat wil zeggen: lees de invoer in variabelen in en voer de berekening uit. Let op, delen door nul bestaat niet. Als de gebruiker probeert te delen door nul is de uitkomst NaN (Not a Number).
Print vervolgens zowel de som als de uitkomst met behulp van het PHP statement **echo**. De som schrijf je in wiskundige notatie, dus + in plaats van "plus" enzovoorts.

c. De website van een internationale on-line drankenhandel, *InterNAT*, is toegankelijk via een HTML formulier waarmee potentiële klanten worden gevraagd hun geboortejaar en hun land van herkomst in te voeren. Voor verschillende landen gelden namelijk andere regels met betrekking tot het aanschaffen en consumeren van alcohol:

land	**minimale leeftijd**
USA	21
UK	18
NL	16
BE	16
RU	12

Schrijf PHP code waarin aan een Boolean variabele $toegang de waarde **true** wordt toegekend als de potentiële klant toegang mag krijgen. In elk ander geval krijgt deze variabele de waarde **false** toegekend. Het geboortejaar van de potentiële klant vindt je in $_POST['*geboortejaar*'] en het land van herkomst in $_POST['*land*'].
Als $toegang geldt, geef dan een welkomsboodschap weer. Geldt de variabele $toegang niet, troost de potentiële klant en vraag hem/haar terug te komen als hij/zij oud genoeg is. Geef daarbij aan hoeveel jaar deze potentiële klant nog moet wachten.

6 Webapplicaties III
Validatie van invoer

6.0 Een webapplicatie om de levensverwachting te berekenen

6.0.0 Rechttoe rechtaan

```
<html>
    <head>
        <title>Bereken je levensverwachting</title>
    </head>
    <body>
        <h2>Bereken je levensverwachting</h2>
        <form action="levensverwachting1.php" method="POST">
            naam: <input type="text" name="naam"><br>
            sekse: man <input type="radio" value="man" name="sekse" checked>
                   vrouw <input type="radio" value="vrouw" name="sekse"><br>
            geboortedatum: <input type="text" size="10"
                                  maxlength="10" name="geboortedatum"
                                  value="dd-mm-jjjj"><br>
            gewicht (kg): <input type="text" name="gewicht"
            size="3"><br>
            opmerkingen:<br>
            <textarea cols="30" rows="5" name="opmerkingen">
            </textarea><br>
            <input type="submit" value="Bereken mijn levensverwachting">
        </form>
        <hr>
    </body>
</html>
```

Listing 6.0: levensverwachting1.html: wat is jouw levensverwachting?

In Listing 6.0 zie je een eenvoudig formulier waarin de gebruiker enkele gegevens in kan vullen waarna zijn of haar levensverwachting wordt berekend door het PHP script in Listing 6.1. De levensverwachting bestaat uit de algemene levensverachting voor mannen of vrouwen in Nederland plus of minus een willekeurig getal tussen 0 en een kwart van het ingevoerde gewicht.

```
<?php
$naam = $_POST['naam'];
$sekse = $_POST['sekse'];
```

```
4  $geboortedatum = $_POST['geboortedatum'];
5
6  // Hak de datum op in stukjes en ken de verschillende onderdelen toe aan
7  // de variabelen dag, maand en jaar.
8  sscanf( $geboortedatum, "%d-%d-%d", $dag, $maand, $jaar);
9
10 $gewicht = $_POST['gewicht'];
11 // Het gewicht speelt een kleine, doch onbekende rol bij de levensverwachting;
12 // deze rol wordt aangegeven door $g_factor:
13 $g_factor = rand( -$gewicht/4, $gewicht/4);
14
15 $opmerkingen = $_POST['opmerkingen'];
16
17 // Bereken de levensverwachting; verschil tussen man en vrouw
18 if ($sekse == 'man') {
19     $levensverwachting = 78 + $g_factor;
20 } else {
21     $levensverwachting = 82 + $g_factor;
22 };
23
24 // Bereken het aantal jaren dat $naam nog te leven heeft:
25 $jaren_over = ($jaar + $levensverwachting) - date('Y');
26 ?>
27 <html>
28     <head>
29         <title>Je levensverwachting</title>
30     </head>
31     <body>
32         <p>Beste <?php echo $naam; ?>,<br>
33         <br>
34         je levensverwachting is <?php echo $levensverwachting; ?>.
35         Je hebt dus nog <?php echo $jaren_over; ?> jaren te leven.
36         </p>
37         <?php
38 if ($opmerkingen == "") {
39 ?>
40         <p><i>Je hebt geen opmerkingen</i></p>
41 <?php
42 } else {
43 ?>
44         <blockquote><?php echo $opmerkingen; ?></blockquote>
45 <?php
46 };
47 ?>
48     </body>
49 </html>
```

Listing 6.1: levensverwachting1.php: bereken de levensverwachting.

6.0.1 En als het bijbehorende formulier niet gebruikt wordt?

Stel nu eens dat je voor jezelf een ander formulier hebt gemaakt om het PHP script aan te roepen? Wat gebeurt er dan? Een voorbeeld van zo'n fout formulier zie je in Listing 6.2.

```
<html>
    <head>
        <title>HACK</title>
    </head>
    <body>
        <h2>HACK</h2>
        <form action="levensverwachting1.php" method="POST">
            naam: <input type="text" name="naam"><br>
            geboortedatum: <input type="text" size="10"
                                  maxlength="10" name="geboortedatum"
                                  value="dd-mm-jjjj"><br>
        </form>
        <hr>
    </body>
</html>
```

Listing 6.2: levensverwachting1a.html: het levensverwachtingsformulier "gehackt".

Een aantal velden, waaronder de gewicht, zitten niet in dit formulier. Het PHP script wordt wel gewoon aangeroepen en het $_POST-array aangemaakt. Maar in dat array zit geen item *'gewicht'*, of *'opmerkingen'*. Het verwerkingsscript probeert die items toch in te lezen. Ogenschijnlijk lijkt het script te werken, maar de uitkomst is niet gebaseerd op de juiste data. Het is jouw taak als programmeur om er voor te zorgen dat het script alleen maar wordt uitgevoerd als alle benodigde gegevens binnen zijn.

Je kunt het script daartoe als volgt aanpassen (Listing 6.3):

```
<?php
if ( isset ($_POST['naam']) and isset($_POST['sekse']) and
     isset ($_POST['geboortedatum']) and isset($_POST['gewicht']) and
     isset ($_POST['opmerkingen'])) {
    // $_POST goed aangemaakt: ga verder met de resultaatpagina

    $naam = $_POST['naam'];
    $sekse = $_POST['sekse'];
    $geboortedatum = $_POST['geboortedatum'];

    sscanf( $geboortedatum, "%d-%d-%d", $dag, $maand, $jaar);

    $gewicht = $_POST['gewicht'];
    $g_factor = rand( -$gewicht/4, $gewicht/4);

    $opmerkingen = $_POST['opmerkingen'];

    if ($sekse == 'man') {
        $levensverwachting = 78 + $g_factor;
    } else {
        $levensverwachting = 82 + $g_factor;
    };

    $jaren_over = ($jaar + $levensverwachting) - date('Y');
?>
<html>
```

```
27      <head>
28          <title>Je levensverwachting</title>
29      </head>
30      <body>
31          <p>Beste <?php echo $naam; ?>,<br>
32          <br>
33          je levensverwachting is <?php echo $levensverwachting; ?>.
34          Je hebt dus nog <?php echo $jaren_over; ?> jaren te leven.
35          </p>
36          <?php
37      if ($opmerkingen == "") {
38  ?>
39          <p><i>Je hebt geen opmerkingen</i></p>
40  <?php
41      } else {
42  ?>
43          <blockquote><?php echo $opmerkingen; ?></blockquote>
44  <?php
45      };
46  ?>
47      </body>
48  </html>
49  <?php
50  } else {
51      // $_POST niet goed aangemaakt
52      // Geef een foutpagina terug
53      echo "Heb je wel gebruik gemaakt van levensverwachtingsformulier?";
54  };
55  ?>
```

Listing 6.3: levensverwachting2.php: zorg ervoor dat alle benodigde gegevens bestaan...

Om te bepalen of het gewicht bestaat gebruiken we dus de functie **isset**($_POST['*gewicht*']): als het gewicht inderdaad bestaat, levert deze expressie de waarde **true** op en anders **false**. Als niet alle gegevens aanwezig zijn, krijgt de gebruiker een kort foutbericht te zien. Let op het lange if-statement: dit loopt door tot aan het einde van het PHP script met halverwege ook nog een else-blok.

6.0.2 En als de gebruiker niets invult?

Ook als alle benodigde gegevens bestaan, hoeven die gegevens niet goed te zijn. Niets weerhoudt de gebruiker ervan niets in te vullen. Om te testen of een variabele "iets" bevat, gebruiken we de functie **empty**($var). Is $var leeg, dus *""* of 0, dan levert deze functie de waarde **true** op. Zie Listing 6.4.

```
1  <?php
2  // Valideer de invoer
3  $foutmeldingen = "";
4  if (! isset ($_POST['naam']) or empty($_POST['naam')) {
5      $foutmeldingen .= "<li>Vul een naam in</li>\n";
6  };
7  if (! isset ($_POST['sekse']) or empty($_POST['sekse')) {
8      $foutmeldingen .= "<li>Vul een sekse in</li>\n";
```

```
9  };
10 if (! isset ($_POST['geboortedatum']) or empty($_POST['geboortedatum'])) {
11     $foutmeldingen .= "<li>Vul een geboortedatum in</li>\n";
12 };
13 if (! isset ($_POST['gewicht']) or empty($_POST['gewicht'])) {
14     $foutmeldingen .= "<li>Vul een gewicht in</li>\n";
15 };
16 if (! isset ($_POST['opmerkingen'])) {
17     $foutmeldingen .= "<li>Gebruik het juiste HTML formulier
18         om je levensverwachting op te vragen.</li>\n";
19 };
20
21 if ($foutmeldingen == "") {
22     // Alles is goedgegaan: maak de resultaatpagina
23     $naam = $_POST['naam'];
24     $sekse = $_POST['sekse'];
25     $geboortedatum = $_POST['geboortedatum'];
26
27     sscanf( $geboortedatum, "%d-%d-%d", $dag, $maand, $jaar);
28
29     $gewicht = $_POST['gewicht'];
30     $g_factor = rand( -$gewicht/4, $gewicht/4);
31
32     $opmerkingen = $_POST['opmerkingen'];
33
34     if ($sekse == 'man') {
35         $levensverwachting = 78 + $g_factor;
36     } else {
37         $levensverwachting = 82 + $g_factor;
38     };
39
40     $jaren_over = ($jaar + $levensverwachting) - date('Y');
41 ?>
42 <html>
43     <head>
44         <title>Je levensverwachting</title>
45     </head>
46     <body>
47         <p>Beste <?php echo $naam; ?>,<br>
48         <br>
49         je levensverwachting is <?php echo $levensverwachting; ?>.
50         Je hebt dus nog <?php echo $jaren_over; ?> jaren te leven.
51         </p>
52         <?php
53     if ($opmerkingen == "") {
54 ?>
55         <p><i>Je hebt geen opmerkingen</i></p>
56 <?php
57     } else {
58 ?>
59         <blockquote><?php echo $opmerkingen; ?></blockquote>
60 <?php
61     };
62 ?>
```

```
63      </body>
64  </html>
65  <?php
66  } else {
67      // Fout bij validatie, geef een foutpagina terug
68  ?>
69  <html>
70      <head>
71          <title>FOUT</title>
72      </head>
73      <body>
74          <h2>Fouten:</h2>
75          <ul>
76              <?php echo $foutmeldingen; ?>
77          </ul>
78      </body>
79  </html>
80  <?php
81  };
82  ?>
```

Listing 6.4: levensverwachting3.php: zorg ervoor dat alle benodigde gegevens bestaan en niet leeg zijn...

In plaats van een kort foutbericht, krijgt de gebruiker een lijst met meer betekenisvolle foutmeldingen te zien. Gaat alles goed, dan genereert PHP gewoon de resultaatpagina. Merk op dat de gebruiker niets bij de opmerkingen hoeft in te vullen ...

6.0.3 Is er wel iets *goeds* ingevuld?

Een volgende stap ter verbetering van de levensverwachtingapplicatie is het controleren of wel *goede* gegevens zijn ingevuld. Zie Listing 6.5 voor enkele voorbeelden. Merk op dat ondanks dat al veel fouten in de invoer opgespoord worden, er nog steeds foute gegevens ingevoerd kunnen worden. Zo kan het gewicht negaties zijn, de datum later dan de huidige datum of een onzinnige naam ingevuld worden.

```
1  <?php
2  // Valideer de invoer
3  $foutmeldingen = "";
4
5  if (! isset ($_POST['naam']) or empty($_POST['naam'])) {
6      $foutmeldingen .= "<li>Vul een naam in</li>\n";
7  } else {
8      $naam = $_POST['naam'];
9  };
10
11 if (! isset ($_POST['sekse']) or empty($_POST['sekse'])) {
12     $foutmeldingen .= "<li>Vul een sekse in</li>\n";
13 } else {
14     $sekse = $_POST['sekse'];
15     if ($sekse == 'man') {
16         $is_man = true;
```

```
17      } elseif ($sekse == 'vrouw') {
18          $is_man = false;
19      } else {
20          $foutmeldingen .= "<li>Voer man of vrouw in bij sekse</li>\n";
21      };
22  };
23
24  if (! isset ($_POST['geboortedatum']) or empty($_POST['geboortedatum'])) {
25      $foutmeldingen .= "<li>Vul een geboortedatum in</li>\n";
26  } else {
27      $geboortedatum = $_POST['geboortedatum'];
28      if (preg_match('/\d\d-\d\d-\d\d\d\d/', $geboortedatum)) {
29          // dit is overigens geen goede check of er wel echt een bestaande
30          // datum is ingevuld ...
31          // Trouwens wat gebeurt er als de ingevulde datum later is dan de
32          // huidige datum?
33          // Valideren is niet zo eenvoudig ...
34          sscanf( $geboortedatum, "%d-%d-%d", $dag, $maand, $jaar);
35      } else {
36          $foutmeldingen .= "<li>Vul een <b>goede</b> datum in volgens
37              het patroon: dd-mm-jjjj, bijvoorbeeld <i>23-09-1978</i>.</li>\n";
38      };
39  };
40
41  if (! isset ($_POST['gewicht']) or empty($_POST['gewicht'])) {
42      $foutmeldingen .= "<li>Vul een gewicht in</li>\n";
43  } else {
44      if (is_numeric($_POST['gewicht'])) {
45          // maak een getal van het gewicht, onduidelijk of int of float
46          $gewicht = $_POST['gewicht'] + 0;
47          if (! is_int ($gewicht)) {
48              $foutmeldingen .= "<li>Voer een geheel getal in, geen kommagetal!</li>\n";
49          };
50      } else {
51          $foutmeldingen .= "<li>Voer een geheel getal in voor het gewicht!</li>\n";
52      };
53  };
54
55  if (! isset ($_POST['opmerkingen'])) {
56      $foutmeldingen .= "<li>Gebruik het juiste HTML formulier om je
57          levensverwachting op te vragen.</li>\n";
58  } else {
59      $opmerkingen = $_POST['opmerkingen'];
60  };
61
62  if ($foutmeldingen == "") {
63      // Alles is goedgegaan: maak de resultaatpagina
64      $g_factor = rand( -$gewicht/4, $gewicht/4);
65      if ($is_man) {
66          $levensverwachting = 78 + $g_factor;
67      } else {
68          $levensverwachting = 82 + $g_factor;
69      };
70
```

```
71      $jaren_over = ($jaar + $levensverwachting) - date('Y');
72  ?>
73  <html>
74      <head>
75          <title>Je levensverwachting</title>
76      </head>
77      <body>
78          <p>Beste <?php echo $naam; ?>,<br>
79          <br>
80          je levensverwachting is <?php echo $levensverwachting; ?>.
81          Je hebt dus nog <?php echo $jaren_over; ?> jaren te leven.
82          </p>
83          <?php
84      if ($opmerkingen == "") {
85  ?>
86          <p><i>Je hebt geen opmerkingen</i></p>
87  <?php
88      } else {
89  ?>
90          <blockquote><?php echo $opmerkingen; ?></blockquote>
91  <?php
92      };
93  ?>
94      </body>
95  </html>
96  <?php
97  } else {
98      // Fout bij validatie, geef een foutpagina terug
99  ?>
100 <html>
101     <head>
102         <title>FOUT</title>
103     </head>
104     <body>
105         <h2>Fouten:</h2>
106         <ul>
107             <?php echo $foutmeldingen; ?>
108         </ul>
109     </body>
110 </html>
111 <?php
112 };
113 ?>
```

Listing 6.5: levensverwachting3.php: de invoer gevalideerd?

Let op het gebruik van de functie **is_numeric**($string) en het truckje om te controleren of een variabele een geheel getal of een kommagetal bevat.

6.1 Vertrouw de invoer niet, nooit

Als je weet dat een PHP script inderdaad via een HTML formulier is aangeroepen, kun je er nog steeds niet van uit gaan dat de gegevens door de gebruiker

goed zijn ingevoerd.[1] Gebruikers kun je niet vertrouwen! Nooit!.

Je kunt gebruikers nog zo goed voorlichten en het formulier nog zo eenvoudig mogelijk maken, maar er is altijd een gebruiker die het toch niet helemaal begrijpt en iets verkeerd invult. Of per ongeluk iets verkeerd invult. Of een gebruiker die op eens gaat nadenken.

Controleer daarom altijd de invoer van de gebruiker. We noemen dit het *valideren* van de gebruikersinvoer. Bij het valideren van gebruikersinvoer test je niet alleen of de invoer wel bestaat, maar ook of de ingevulde waarde aan bepaalde eisen voldoet. Of, anders gezegd, slaat de invoer ergens op?

Stel je hebt een webapplicatie waarbij de gebruiker zijn leeftijd moet invullen. Een ingevulde leeftijd is goed als het een geheel getal is tussen 0 en 150. Alle andere waarden zijn fout.

Een ander voorbeeld: in een webapplicatie om schoolrapporten uit te rekenen, moeten ingevulde punten altijd tussen 1 en 10 inliggen. Leerlingnummers voldoen aan een bepaald patroon. Namen van leerlingen moeten in de schooladminsitratie voorkomen. Klassen moeten bestaan. Enzovoorts.

PHP kent een aantal handige functies die je bij het valideren van invoer kunt gebruiken. De functie `empty( $var )` heb je al gezien:

functie	**omschrijving**
`isset ( $var )`	Controleert of `$var` überhaupt bestaat
`empty( $var )`	Controleert of `$var` leeg is
`is_int ( $var )`	Controleert of `$var` een geheel getal is
`is_float ( $var )`	Controleert of `$var` een komma getal is
`is_bool ( $var )`	Controleert of `$var` een waarheidwaarde is
`is_string ( $var )`	Controleert of `$var` een string is
`is_array ( $var )`	Controleert of `$var` een array is
`is_numeric( $string )`	Controleert of `$string` omgezet kan worden naar een getal (zowel integer als float)

6.2 Standaardschema

Valideren doe je met behulp van het if-statement. Als een fout in de invoer ontdekt wordt, wil je de gebruiker daarover informeren. En je wilt de gebruiker over *alle* fouten informeren. Het is handig als je de foutmeldingen bijhoudt in een variabele. Als er fouten zijn, geef je die inhoud van die variabele weer.

Een PHP script met validatie bestaat altijd uit een if-statement om te controleren of `$_POST` bestaat. In dat if-statement schrijf je een if-statement voor elk ingevoerd gegeven. Na de validatie controleer je of er fouten zijn gemaakt. Zo nee: verwerk de ingevoerde gegevens en toon de resultaatpagina. Zo ja: toon de foutpagina.

Je PHP script bouw je dus eigenlijk altijd op dezelfde manier op. Bekijk Listin 6.5 nog maar eens. In het kort:

1. Maak de lege string variabele `$foutmeldingen` aan.

2. Voor elke ingevoerde waarde:

[1] Of dat de binnenkomende gegevens überhaupt door een gebruiker zijn ingevuld en niet door een hacker, een cracker of een of ander programma. Nogmaals, vertrouw niets dat van het internet afkomt.

- Bestaat de waarde überhaupt? **isset** ()
- Moet de waarde ingevoerd zijn? **empty**()
- Welke extra eisen worden aan de waarde gesteld? Een goede datum, een positief getal, een getal tussen 1 en 10, enzovoorts.

3. Als de variabele met foutmeldingen nog steeds leeg is voert PHP eventuele berekeningen uit en maakt de resultaatpagina. Zijn er foutmeldingen? Maak dan de foutpagina aan. Dit is een *lang* if-statement met daartussen HTML code.

6.3 Praktische opgaven

6.3.0 Pas de PHP scripts die je in vorige hoofdstukken hebt gemaakt zodanig aan dat de gebruikersinvoer gevalideerd wordt.

a. Pas het PHP script dat je bij opgave 4.0.1 hebt gemaakt zo aan dat gecontroleerd wordt dat de gebruiker een positief getal invoert voor de straal.
b. Pas het PHP script `leeftijd.php` aan dat je bij opdracht 1.0.2 hebt gemaakt. De ingevoerde naam mag niet leeg zijn en het geboortejaar moet een geheel getal tussen 1900 en 2009 zijn. Zorg dat PHP de ingevoerde gegevens valideert.
c. Pas het PHP script `boekfactuur.php` aan dat je bij opdracht 4.0.0 hebt gemaakt. Zorg ervoor dat de klant alle gegevens goed invoert. Dat wil zeggen dat het aantal een geheel getal is, zijn NAW gegevens aanwezig en een betaalwijze aangevinkt, enzovoorts.

⋆ **6.3.1** Herschrijf je antwoorden op de vragen uit het vorige hoofdstuk zodanig dat alle invoer ook gevalideerd wordt.

7 Arrays en foreach

7.0 Arrays

7.0.0 Associatieve arrays

Bij het maken van webapplicaties ben je het associatieve array $_POST al regelmatig tegengekomen. In dat array zijn de waarden te vinden die de gebruiker in het HTML formulier heeft ingevuld. Je kunt de waarden uit dit array opvragen met behulp van een sleutel: de naam van het control in het HTML formulier.

PHP is niet de enige die associatieve arrays aan kan maken, dat kun je ook zelf in je HTML formulieren:

```
1  ...
2  <form action="..." method="POST">
3      <fieldset>
4          <legend>Vul leerlinggegevens in:</legend>
5          leerlingnummer: <input type="text" name="leerling[llnr]"><br>
6          voornaam: <input type="text" name="leerling[voornaam]"><br>
7          tussenvoegsel: <input type="text" name="leerling[tussenvoegsel]"><br>
8          achternaam: <input type="text" name="leerling[achternaam]"><br>
9          geboortejaar: <input type="text" name="leerling[geboortejaar]">
10     </fieldset>
11     <input type="submit" value="verstuur">
12 </form>
13 ...
```

In je verwerkingsscript lees je het leerling array als volgt in om het daarna te gebruiken:

```php
<?php
  // Je leest eerst het hele array in een keer in:
   $leerling = $_POST['leerling'];

  // Vervolgens kun je het array gebruiken: je kunt waarden
  // opvragen, aanpassen, gebruiken en toevoegen:

  // opvragen en gebruiken:
   $leeftijd = date( 'l' ) - $leerling [ 'geboortejaar' ];

  // nieuwe waarde met sleutel toevoegen:
   $leerling [ 'klas' ] = "brug1";

  // bestaande waarden aanpassen:
   $leerling [ 'geboortejaar' ] = 1990;

  // waarden gebruiken:
  echo $leerling [ 'voornaam' ] . " is geboren in " . $leerling [ 'geboortejaar' ];
?>
```

Dat leerling array kun je ook in PHP code maken. Je moet dan wel zelf, als programmeur, waarden geven:

```php
<?php
// associatief array aanmaken:
 $leerling = array(
   'llnr' => 23434,
   'voornaam' => 'Jan',
   'tussenvoegsel' => '',
   'achternaam' => 'Cornelisz',
   'geboortejaar' => 1991 );
?>
```

associatief array

```
$assoc_arr = array(
   sleutel_1 => waarde_1,
   sleutel_2 => waarde_2,
   //          ...
   sleutel_N => waarde_N );
```

De sleutels zijn strings of gehele getallen, de waarden mogen elk type hebben. Je kunt vervolgens de waarden in het array opvragen of aanpassen met behulp van de sleutel tussen rechte haken achter de variabele te plaatsen: $assoc_arr [sleutel] levert de waarde die bij deze sleutel hoort. Met behulp van een toekenning kun je aan een associatief array nieuwe sleutels met waarden toevoegen.

7.0.1 Gewone arrays

Zo'n associatief array aanmaken is veel werk: je moet voor elke waarde een zinvolle sleutel bedenken. Soms wil je gewoon een lijst met waarden aanmaken waarbij sleutels helemaal geen extra betekenis hebben. Je maakt dan een *gewoon array* aan.

array

In tegenstelling tot een associatief array geef je bij een gewoon array geen sleutels aan. Je maakt een gewoon array als volgt:
$arr = **array**(waarde_1, waarde_2, ..., waarde_N);.

PHP maakt automatisch sleutels aan voor de waarden in het array. We noemen deze automatisch aangemaakte sleutels *indices*. De eerste waarde krijgt index 0, de volgende waarde index 1 en de laatste waarde index $N - 1$. PHP begint dus bij 0 te tellen en niet bij 1!

Je kunt waarden in het array opvragen, aanpassen en toevoegen door de index tussen rechte haken achter de variabele naam te plaatsen.

PHP kent een hele hoop functies om met arrays werken:
http://nl3.php.net/manual/en/book.array.php.

Ook gewone arrays kun je in je HTML formulieren aanmaken. Stel je wilt je eindcijfer voor het vak Informatica uitrekenen. Er zijn maar vier toetsen, dus je maakt een HTML formulier met vier tekstvakjes en een knop:

```
...
<form action="bereken_eindcijfer.php" method="POST">
  <p>
    Voer de punten voor de vier toetsen hieronder in:<br>
    <input type="text" name="punten[]"><br>
    <input type="text" name="punten[]"><br>
    <input type="text" name="punten[]"><br>
    <input type="text" name="punten[]"><br>
    <input type="submit" value="Bereken eindcijfer">
  </p>
</form>
...
```

Je eindcijfer bereken je dan als volgt met PHP:

```
<?php
$punten = $_POST['punten'];
// Let op, indexen beginnnen bij 0
 $eindcijfer  = ($punten[0] + $punten[1] + $punten[2] + $punten[3]) / 4;
echo "Je eindcijfer is: $eindcijfer ";
?>
```

7.1 Het foreach-statement

Dit werkt. Toch zijn er een aantal nadelen. Je schrijft heel veel vergelijkbare code waarin je veel fouten kunt maken. En wat als je voor een vak meer dan vier

punten kunt scoren, dan wordt de berekening alleen maar langer. De computer was er toch om ons werk uit handen te nemen?

Belangrijker is echter het bezwaar dat je als programmeur precies moet weten hoeveel elementen er in het array zitten. Immers je haalt de waarden uit het array met behulp van de indices en je deelt door het aantal elementen in het array. Maar wat nu als je een punt vervalt of er een bij komt? Dan moet je dat in al je code weer gaan aanpassen.

Dat kan handiger:

```
1  <?php
2  $punten = $_POST['punten'];
3
4  $som = 0;
5  foreach ( $punten as $punt ) {
6    $som += $punt;
7  }
8
9   $eindcijfer  = $som / count( $punten );
10 echo "Je eindcijfer is $eindcijfer ";
11 ?>
```

Deze code is een stuk overzichtelijk en bruikbaar voor elk array met punten, ongeacht het aantal punten. In dit voorbeeld worden twee nieuwe PHP elementen gebruikt: de functie **count**($arr) en het statement **foreach**.

De PHP functie **count**($arr) geeft als waarde het aantal elementen in het array $arr terug. Het foreach-statement is een zogenaamd *herhalings*-statement.

foreach-statement

Er zijn twee varianten van het foreach-statement:

0.
```
foreach ( $arr as $waarde ) {
    // doe iets met $waarde,
    // voor alle waarden in $arr
    echo $waarde;
};
```

1.
```
foreach ( $arr as $sleutel => $waarde )
    // doe iets met $sleutel en/of $waarde,
    // voor alle sleutels en waarden in $arr
    echo "Bij sleutel " . $sleutel . " hoort waarde " . $waarde;
};
```

Het foreach-statement voert de PHP statements in het blok na het foreach-statement uit voor elke waarde, of voor elke sleutel en bijbehorende waarde, in het array $arr.

Voordat het blok wordt uitgevoerd, krijgen de variabelen $sleutel en $waarde als nieuwe waarde de volgende sleutel en waarde uit het array automatisch toegekend. Die variabelen met bijbehorende waarde zijn dus in het blok bekend en kun je gewoon gebruiken zoals elke andere variabele.

Let op: pas het array in het foreach-statement niet aan. Zie voor meer informatie: http://www.php.net/manual/en/control-structures.foreach.php.

7.2 Aankruisvakjes en meervoudige keuzelijsten

In een HTML formulier kun je ook aankruisvakjes opnemen. De gebruiker kan verschillende opties tegelijk selecteren. Al deze geselecteerde opties horen bij echter bij elkaar: al deze aankruisvakjes hebben dezelfde naam. Om ook de geselecteerde waarden bij elkaar te houden, gebruik je een array:

```
...
<form action="..." method="POST">
    <input type="checkbox" name="fruit[]" value="elstar">Elstar<br>
    <input type="checkbox" name="fruit[]" value="jonagold">Jona Gold<br>
    <input type="checkbox" name="fruit[]" value="granny">Granny<br>
    <input type="checkbox" name="fruit[]" value="conference">Conference<br>
    <input type="checkbox" name="fruit[]" value="fuji">Fuji<br>
    <input type="checkbox" name="fruit[]" value="stoof">stoof<br>

    <input type="submit" value="verstuur">
</form>
...
```

Op vergelijkbare wijze kun je de geselecteerde items uit een meervoudige keuzelijst opvragen en gebruiken.

7.3 Gemenge Opgaven

7.3.0

Gegeven is de array $veel_getallen met daarin heel veel getallen. Schrijf PHP code (zonder PHP te gebruiken) waarin je de som (alle getallen bij elkaar opgeteld) en het product (alle getallen met elkaar vermenigvuldigd) van alle getallen in het array $veel_getallen berekent en weergeeft. Maak gebruik van het foreach-statement.

Je hoeft $veel_getallen niet zelf aan te maken: die lees je in vanuit het $_POST array. Er is een HTML formulier waarin heel veel getallen ingevuld kunnen worden. Test vervolgens je PHP script.

7.3.1 Bekijk het volgende stukje HTML code:

```
...
<form action="stemming.php" method="POST">
  <p>
    Vul het aantal stemmen in:<br>
    heeft 'ja' gestemd: <input type="text" name="stemmen[ja]"><br>
    heeft 'nee' gestemd: <input type="text" name="stemmen[nee]"><br>
    heeft 'weet niet' gestemd: <input type="text" name="stemmen[weetniet]"><br>
    heeft ongeldig gestemd: <input type="text" name="stemmen[ongeldig]"><br>

    <input type="submit" value="Bereken percentages">
  </p>
</form>
...
```

Schrijf PHP code, zonder PHP te gebruiken, om de percentages van de stemming

te berekenen. Er is gegeven dat de gebruiker alles correct invult, je hoeft de invoer dus niet te valideren. Maak gebruik van het foreach-statement.

Opmerking: de gebruiker vult het totaal aantal ja-stemmen, het totaal aantal nee-stemmen, het totaal aantal weet-niet-stemmen en het totaal aantal ongeldige stemmen in. Dus niet een stem per keer ...

Geef de percentages samen met het ingevoerde aantal weer in een HTML tabel. Een voorbeeld van zo'n tabel is:

```
<table border="1">
  <tr><th>stem</th><th>aantal</th><th>percentage</th></tr>

  <tr><td>ja</td><td>23</td><td>28%</td></tr>
  <tr><td>nee</td><td>45</td><td>55%</td></tr>
  <tr><td>weet niet</td><td>12</td><td>14.6%</td></tr>
  <tr><td>ongeldig</td><td>2</td><td>2.4%</td></tr>

  <tr><th>totaal</th><td>82</td><td>100%</td></tr>
</table>
```

Jouw PHP code *genereert* dus zo'n HTML tabel gebaseerd op de door de gebruiker ingevoerde gegevens.

Test vervolgens je PHP script.

7.3.2 Bekijk het volgende stukje HTML code:

```
...
<form action="puntenlijst.php" method="POST">
  <p>
    Voer hieronder een aantal punten in<br>

    <input type="text" name="punten[]"><br>
    <input type="text" name="punten[]"><br>
    <input type="text" name="punten[]"><br>
    <input type="text" name="punten[]"><br>
    <input type="text" name="punten[]"><br>
    <input type="text" name="punten[]"><br>
    <input type="text" name="punten[]"><br>
    <input type="text" name="punten[]"><br>
    <input type="text" name="punten[]"><br>
    <input type="text" name="punten[]"><br>

    <input type="submit" value="Bereken gemiddelde">
  </p>
</form>
...
```

Let op, de gebruiker hoeft niet alle tien de tekstvakjes in te vullen. Met andere woorden, je weet van te voren niet hoeveel vakjes de gebruiker invult, de gebruiker kan nul punten, een punt, acht punten, enzovoorts invullen. De functie **count**($punten) kun je dus niet gebruiken om het aantal ingevulde punten te bepalen.

a. Schrijf PHP code (zonder PHP te gebruiken) om het gemiddelde van de ingevulde punten te berekenen en weer te geven. Er is gegeven dat de gebruiker,

als hij iets invult, getallen invoert tussen 1 en 10 (dat controleren we in de volgende deelopgave). Maak gebruik van het foreach-statement.

b. $\star$ Schrijf PHP code (zonder PHP te gebruiken) waarin je de invoer van de gebruiker valideert. Dat wil zeggen, als de gebruiker iets in een tekstvakje invult, dan moet dat een getal zijn tussen 1 en 10. Het is immers een punt. Als de gebruiker niets in een tekstvakje invult, is dat ook goed. Je mag aannemen dat de variabele $_POST bestaat. Maak gebruik van het foreach-statement.
Stel de gebruiker vervolgens op de hoogte als hij een fout heeft gemaakt bij het invoeren van punten.

8 Functies

deel je programma op in kleinere stukken

8.0 Het gebruik van functies

Je hebt al een aantal PHP functies gebruikt: **date**(*'l'*) om de dag van de week op te vragen, **pi**() om π uit te rekenen, enzovoorts. PHP kent nog honderden andere functies. Op http://www.php.net/manual/en/funcref.php vind je een functiereferentie van heel veel functies in PHP.

Deze lijst kent functies om met arrays te werken, functies om databases aan te spreken, functies om afbeeldingen te maken en functies om e-mail te sturen. Voor bijna elke situatie is wel een passende functie. Het enige probleem: vindt de meest geschikte functie.

Er zijn twee verschillende soorten functies: functies die een waarde retourneren, zoals **pi**() en **sin**(0.462), en functies die *geen* waarde retourneren maar "iets" doen, zoals **var_dump**($var) dat de waarde en het type van $var afdrukt. Functies die geen waarde retourneren, vormen een PHP statement op zich en je schrijft ze dus alleen op een regel afgesloten met een punt-komma.

Functies die wel een waarde retourneren zijn *altijd* expressies en staan dus aan de *rechterkant* van het =-teken *of* maken deel uit van een conditie.

```
1  <?php
2  // functies die waarden retourneren staan aan de rechterkant van het =-teken
3  $r = sin( 0.453 );
4  $opp = 2 * pi() * $r;
5  // of maken deel uit van een conditie :
6  if (is_numeric( $_POST['getal'] ) ) {
7    $getal = $_POST['getal'];
8    if ( cos( $getal ) <= 0.5 or pow( $getal, 24 ) < 1296885 ) {
9      // functies die geen waarde retourneren staan op zichzelf op
10     // een regel : ze zijn een PHP statement.
11     var_dump( $getal );
12     exit ();
13   };
14 };
15 ?>
```

Functies hebben nul of meer *argumenten* of *parameters* tussen ronde haakjes. Daarmee stuur je informatie *naar* de functie. De functie verwerkt vervolgens die informatie. Zo rekent de functie **sin**(0.453) de sinus van het getal 0.453 uit. Hoe die functie verder werkt, is niet van belang. In de *functiedefinitie* is

vastgelegd dat als de invoer een getal is, dat de uitvoer dan de sinus van dat getal is.

Je bent er als programmeur zelf verantwoordelijk voor dat je aan de functies die je gebruikt de waarden en variabelen van het juiste type meegeeft.

8.1 Het maken van functies

Naast de vele PHP functies die je zo kunt gebruiken, kun je ook zelf een functie maken en vervolgens gebruiken. Je definieert daartoe eerst de functie. Eenmaal gedefiniëerd, kun je de functie gebruiken. Functiedefinities schrijf je dus *bovenaan* je PHP scripts; in je PHP code daaronder kun je je eigen functies vervolgens gebruiken.

functiedefinitie

```
function functie_naam( $arg_1, $arg_2, ..., $arg_n ) {
  // doe iets met de argumenten 1 tot en met n en bereken een waarde
  ...
  // en retourneer die waarde
  return waarde;
};
```

Een functie definieer je met het keyword function gevolgd door de functienaam. Daarna schrijf je lijst met argumenten tussen ronde haakjes. Die lijst mag bestaan uit nul argumenten, maar de ronde haakjes schrijf je *altijd* na de functienaam. Tot slot volgen de PHP statements die de functie uitvoert tussen accolades.

Als de functie een waarde retourneert, bevat de functie een of meerdere *return*-statements: het keyword return gevolgd door een expressie. Als de functie geen waarde retourneert, dan bevat de functie *geen* return-statement.

Enkele voorbeelden:

- een functie om “Hallo!” te zeggen: geen argumenten, geen returnwaarde

```
1 function hallo() {
2   echo "Hallo!";
3 };
```

- een functie om iemand bij naam te begroeten: een argument, de naam en geen returnwaarde

```
1 function hallo( $naam ) {
2   echo "Hallo $naam!";
3 };
```

- een functie om het kwadraat van een getal te berekenen:, een argument en een returnwaarde

```
1 function kwadraat( $x ) {
2   return $x * $x;
3 };
```

- een functie om het gemiddelde van een array met getallen te berekenen:

```
function array_gemiddelde( $array ) {
  $totaal = 0;
  foreach( $array as $getal ) {
    $totaal += $getal;
  };
  return $totaal / count( $array );
};
```

- een functie om het even-zijn van een getal te bepalen:

```
function even( $getal ) {
  return ($getal % 2 == 0);
};
```

8.2 Waarom functies? Een uitgewerkt voorbeeld

Vervolgens kun je deze functies in je eigen code gebruiken. Je code wordt er een stuk overzichtelijker en leesbaarder van. Maar wanneer maak je een functie? Op het moment dat je merkt dat vaker hetzelfde stukje code schrijft, is het verstandig een functie te schrijven die dat stukje code uitvoert.

Bijvoorbeeld bij het valideren van invoer gebruik je keer op keer dezelfde code om te controleren of de gebruiker een veld heeft ingevoerd. Je kunt daarvoor dan een functie introduceren.

Een andere reden om een functie te maken is om lange stukken code te vermijden: die zijn onoverzichtelijk en daardoor foutgevoelig. Hak in dat geval de lange code op in kleinere logische stukken en maak daar functies van. Zorg er dan wel voor dat die functies maar een ding berekenen. Het is dan een stuk eenvoudiger om het programma te onderhouden.

Een voorbeeld: een PHP script dat automatisch een matrix met getallen (een array van arrays) weergeeft in een HTML tabel waarbij de rijen in de tabel om en om een andere achtergrondkleur hebben. De matrix wordt in dit voorbeeld in de code aangemaakt. Normaal gesproken halen we zoiets uit een database of een extern bestand.

```
<?php
function is_even( $getal ) {
  // Is $getal even?
  return ($getal % 2 == 0);
};

function tabelrij ( $rij_nr , $rij_array ) {
  // Maak een mooie HTML tabelrij van de gegevens in $rij_array . Als
  // het $rij_nr even is , krijgt de HTML rij een andere class .
  // De HTML tabelrij wordt als een string teruggestuurd

  $resultaat = "";

  if ( is_even ( $rij_nr )) {
    $resultaat .= '<tr class="even">';
```

```
16    } else {
17      $resultaat .= '<tr class="oneven">';
18    };
19
20    foreach( $rij_array as $veld ) {
21      $resultaat .= "<td>$veld</td>";
22    };
23
24    return $resultaat . "</tr>";
25  };
26
27  // In $gegevens maken we een matrix met getallen aan
28  $gegevens = array(
29    array( 12, 23, 172, 234, 1475, 1235, 153),
30    array( 2, 253, 122, 234, 1645, 1623, 15),
31    array( 152, 3, 2, 2634, 145, 1635, 13),
32    array( 172, 2, 2, 234, 145, 12, 153),
33    array( 612, 3, 1, 24, 145, 1625, 153),
34    array( 512, 523, 2, 23, 15, 1235, 13),
35    array( 127, 723, 1, 4, 1645, 135, 153)
36  );
37  ?>
38  <html>
39    <head>
40      <title>Tabelvoorbeeld: het gebruik van functies</title>
41      <style>
42        .even { background-color: Khaki; }
43        .oneven {background-color: Olive; }
44      </style>
45    <head>
46    <body>
47      <h2>Tabelvoorbeeld: het gebruik van functies</h2>
48      <table>
49  <?php
50  foreach( $gegevens as $index => $rij ) {
51    echo "\t\t\t" . tabelrij ( $index, $rij ) . "\n";
52  };
53  ?>
54      </table>
55    </body>
56  </html>
```

Als je de twee functies in een apart PHP bestand opslaat, bijvoorbeeld `tabelfuncties.php`, dan wordt het voorbeeld nog korter en overzichtelijker:

```
1  <?php
2  // We lezen onze functies in:
3  require( "tabelfuncties.php" );
4
5  // In $gegevens maken we een matrix met getallen aan.
6  $gegevens = array(
7    array( 12, 23, 172, 234, 1475, 1235, 153),
8    array( 2, 253, 122, 234, 1645, 1623, 15),
9    array( 152, 3, 2, 2634, 145, 1635, 13),
```

```
10    array( 172, 2, 2, 234, 145, 12, 153),
11    array( 612, 3, 1, 24, 145, 1625, 153),
12    array( 512, 523, 2, 23, 15, 1235, 13),
13    array( 127, 723, 1, 4, 1645, 135, 153)
14  );
15  ?>
16  <html>
17    <head>
18      <title>Tabelvoorbeeld: het gebruik van functies</title>
19      <style>
20        .even { background-color: Khaki; }
21        .oneven {background-color: Olive; }
22      </style>
23    <head>
24    <body>
25      <h2>Tabelvoorbeeld: het gebruik van functies</h2>
26      <table>
27  <?php
28  foreach( $gegevens as $index => $rij ) {
29    echo "\t\t" . tabelrij ( $index, $rij ) . "\n";
30  };
31  ?>
32      </table>
33    </body>
34  </html>
```

Je kunt zo een eigen *bibliotheek* met eigengemaakte functies aanleggen en keer op keer gebruiken. Al het "zware PHP werk" zit dan in die functies verborgen; de resultaatpagina's zijn (weer) eenvoudig van opzet.

8.3 Gemengde opgaven

8.3.0 Schrijf een functie is_punt($getal) dat, gegeven een getal (het argument $getal) **true** retourneert als dat getal kleiner of gelijk aan tien is en groter of gelijk aan een. In elk ander geval retourneert deze functie de waarde **false**.

8.3.1 Schrijf de functie **max**($getal_1, $getal_2) dat het grootste getal van de getallen $getal_1 en $getal_2 retourneert. Als beide getallen even groot zijn, maakt het niet uit welk getal de functie retourneert.

8.3.2 Schrijf de functie **max**($array) dat het grootste getal uit het array $array retourneert. Je roept bij deze opgave de functie **max**($getal_1, $getal_2) uit de vorige opgave aan.

8.3.3 Schrijf de functie steeds_groter($arrray). De functie retourneert **true** als elk opeenvolgend element in het array $array groter is dan het vorige. Als dat niet zo is, retourneert de functie **false**. We nemen aan dat het eerste element altijd groter is dan het (niet bestaande) vorige element.

9 Repetitie

PHP herhaalt

De eerste repetitie heb je al gezien: het foreach-statement om elementen in een array af te lopen. Naast code uitvoeren voor alle elementen in een array, zijn er andere situaties waarin je PHP code vaker wilt herhalen. Er zijn twee herhalings-statements: het while-statement en het for-statement. We noemen zo'n herhaling ook wel een repetitie of een *loop*.

9.0 Het for-statement

PHP kent de pow($grondtal, $exponent) functie om de macht g^e uit te rekenen, met g het grondtal en e de exponent. Als je een macht wil uitrekenen, is het verstandig die functie te gebruiken, maar als een voorbeeld van het gebruik van het for-statement gaan we nu zelf een machtfunctie maken.

Wat is een macht? Er zijn drie gevallen: de exponent is negatief, de exponent is nul en de exponent is positief. Het geval dat de exponent gelijk is aan nul is eenvoudig: $g^0 = 1$, voor alle g. Zoals je weet is g^e gelijk aan

$$\underbrace{g \times g \times g \times \cdots \times g}_{\text{e keer}}$$

(voor positieve e). g^{-e}, Tot slot, is gelijk aan $\frac{1}{g^e}$.

Voor de duidelijkheid van het voorbeeld maken we een machtfunctie die alleen met exponenten ≥ 0 werkt:

```
1  function macht( $g, $e ) {
2    // werkt voor $e >= 0
3
4    $macht = 1; // $g^0 is immers 1, voor alle $g
5
6    for( $i=1; $i <= $e; $i++ ) {
7      $macht *= $g;
8    };
9
10   return $macht;
11 };
```

De machtsfunctie heeft twee argumenten, $g en $e, voor grondtal en exponent en retourneert een waarde, de macht $\$g^{\$e}$. Op regels 6–8 staat het for-statement. In het for-statement wordt $g vermenigvuldigd met $macht en het resultaat die vermenigvuldiging wordt toegekend aan de variabele $macht. Het for-statement

zorgt ervoor dat die vermenigvuldiging herhaaldelijk plaatsvindt, namelijk $e keer.

Het e keer laten herhalen, geef je aan met de eerste regel van het for-statement: je introduceert een tellertje $i die de *beginwaarde* 1 krijgt. Het tellen moet ophouden wanneer het tellertje de waarde van $e heeft bereikt. Tot slot verhoog je iedere stap het tellertje met een. Deze drie onderdelen, *initialisatie*, *einde* en *voortgang* schrijf je altijd op dezelfde manier op:

for-statement

```
for ( initialisatie ; einde; voortgang) {
  // voer deze code herhaaldelijk uit

};
```

Vaak gebruik je het for-statement om te tellen. Het for-statement ziet er dan zo uit:

```
for( $i = begin; $i <= eind; $i++ ) {
  // voer deze code eind - begin keer uit

};
```

Voor het tellertje gebruiken we vaak i of j.

9.1 Het while-statement

Net zoals het for-statement gebruiken we het while-statement om een stukje code herhaaldelijk uit te voeren. Waar het for-statement nadrukkelijk bedoeld is om te tellen, is het while-statement een stuk flexibeler. Hetzelfde voorbeeld met behulp van een while-statement:

```
1  function macht( $g, $e ) {
2    // werkt voor $e >= 0
3    $macht = 1;
4    $i = 1;
5    while ($i <= $e) {
6      $macht *= $g;
7      $i++;
8    };
9    return $macht;
10 };
```

Zoals je ziet verschilt de machtsfunctie met het while-statement amper van de machtsfunctie met het for-statement. Ook nu maken we gebruik van een tellertje. We geven die teller nu een beginwaarde vóór het while-statement. Het while-statement heeft nu een conditie (net zoals bij het if-statement tussen ronde haken) dat het einde van de while-loop aangeeft. Het ophogen van de teller gebeurt in de while-loop.

while-statement

```
while ( conditie ) {
  // Voer deze code uit net zolang als de conditie geldt.

};
```

Zolang de conditie geldt, voert het while-statement voert de PHP code tussen de accolades uit. Na elke *loop* controleert PHP opnieuw of de conditie geldt. Zo ja, dan wordt de loop nog eens uitgevoerd, zo neen, dan gaat PHP verder op de eerste regel na het while-statement.

Let op: zorg ervoor dat je in de loop de conditie ooit ongeldig maakt. Doe je dat niet, dan blijft PHP voor altijd en eeuwig het while-statement uitvoeren. We zeggen dan dat PHP in een *oneindige lus* is terechtgekomen.

Als je een stuk code een vast aantal stappen wilt uitvoeren, is het for-statement een stuk geschikter dan het while-statement. Het for-statement is daar namelijk voor gemaakt. Dat wil niet zeggen dat het while-statement niet werkt. Sterker nog, bij veel problemen maakt het niet zoveel uit welk statement je gebruikt.

9.2 Gemengde opgaven

9.2.0 Schrijf een functie fac($x) dat gegeven een invoergetal $x de faculteit van x retourneert. De faculteit van x, geschreven als $x!$, is gelijk aan: $1 \times 2 \times 3 \times 4 \times \cdots \times x$. Je kunt niet de faculteit van een negatief getal uitrekenen. De faculteit van 0 is gelijk aan 1. Maak gebruik van het for-statement.

9.2.1 De functie is_even($array) is als volgt gedefiniëerd: als alle getallen in het array $array even zijn, retourneert de functie **true**. In elk ander geval retourneert de functie **false**. Maak deze functie met behulp van het while-statement. Zodra een oneven getal gevonden wordt, kan met de while-loop gestopt worden.

9.2.2 Schrijf de functie vind_in($array, $waarde) met behulp van een while-statement. Deze functie retourneert de index van het eerste element in $array met waarde gelijk aan $waarde. Als de waarde niet in het array voorkomt, retourneert de functie de waarde -1.

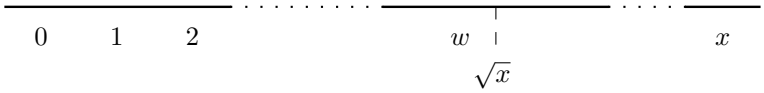
9.3 Extra opgaven

9.3.3 Schrijf een machtsfunctie zonder gebruik te maken van de functie pow() of een iteratiestatement. Deze machtsfunctie kan ook omgaan met een negatieve exponent.

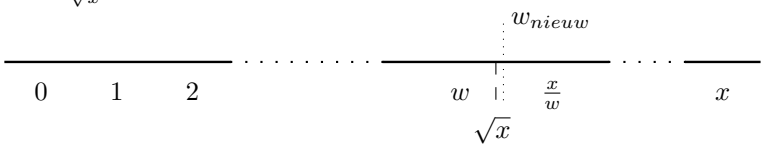
9.3.4 Er zijn verschillende methoden om de wortel van een getal uit te rekenen (kijk maar eens op http://en.wikipedia.org/wiki/Methods_of_computing_square_roots).

Een eenvoudig algoritme is de Babylonische methode:

stap 0 kies een getal w zo dicht mogelijk in de buurt van $\sqrt{x}$. w is een *benadering* van $\sqrt{x}$. De getallenlijn ziet er dan ongeveer zo uit:



stap 1 Voeg $\frac{x}{w}$ aan de getallenlijn toe. Omdat w een benadering is van $\sqrt{x}$ geldt dat $w * \frac{x}{w}$ ongeveer gelijk is aan x en dat $\sqrt{x}$ tussen w en $\frac{x}{w}$ in ligt. (Immers $\sqrt{x} * \frac{x}{\sqrt{x}} = x$).



Het midden tussen w en $\frac{x}{w}$ is een betere benadering van $\sqrt{x}$. Noem deze nieuwe benadering w_{nieuw}. Met andere woorden, w_{nieuw} krijgt de waarde $\frac{w+\frac{x}{w}}{2}$.

stap 2: Zolang w_{nieuw} geen goede benadering van $\sqrt{x}$ is, geef w de waarde van w_{nieuw} en herhaal **stap 1**.

w_{nieuw} is een goede benadering wanneer het verschil tussen w_{nieuw} en $\sqrt{x}$ voldoende klein is, bijvoorbeeld 0.00001 of nog kleiner. Zo'n klein getal noemen we *epsilon*, of kortweg ϵ. Met andere woorden, als $(w_{nieuw})^2 - x \leq \epsilon$ *én* $(w_{nieuw})^2 - x \geq -\epsilon$, dan is w_{nieuw} een goede benadering voor $\sqrt{x}$.

stap 3: w_{nieuw} is een goede benadering van $\sqrt{x}$.

Maak een eenvoudig HTML formulier om een getal in te voeren waaruit je PHP script vervolgens de wortel trekt na drukken op de submitknop. implementeer het hier boven beschreven algoritme in je rekenmachine met behulp van een **while**-loop.

a. Wat is een goede beginkeuze voor w?
b. Welke constanten worden gebruikt?
c. Wat is de conditie van de **while**-loop?
   tip: kijk eens naar de PHP functie **abs**($x)
   (Zie: http://www.php.net/manual/en/function.abs.php).
d. Welke variabelen krijgen een nieuwe waarde in de **while**-loop? Wat zijn hun initiële waarden? Experimenteer met verschillende waarden voor ϵ. Wat is het verschil?
e. Is jouw implementatie van worteltrekken beter of slechter dan die van je grafische rekenmachine?
f. Herschrijf de **while**-loop zodat de conditie ervan een enkele Boolean variabele is. Waarom zou je een **while**-loop zo herschrijven?
g. Worteltrekken werkt alleen voor getallen groter of gelijk aan 0. Zorg ervoor dat de gebruiker niet de wortel van een negatief getal kan trekken.

9.3.5

Zoek een algoritme voor het berekenen van de derdemachtswortel en implementeer dat algoritme als een PHP functie.

tip: Het is eenvoudiger om een algoritme te implementeren als dat algoritme al in een of andere programmeertaal is uitgewerkt, zelfs als dat niet PHP is.

9.3.6 Schrijf de functie vind_in ($array , $waarde) met behulp van een while-statement. Deze functie retourneert de index van het element in $array met waarde gelijk aan $waarde. Als de waarde niet in het array voorkomt, retourneert de functie de waarde -1.

Er is gegeven dat het array oplopend gesorteerd is. Dat wil zeggen dat elk opeenvolgend element in het array groter of gelijk is aan het voorgaande element. Maak deze functie door een zogenaamde *binary search* te implementeren. Zie: http://en.wikipedia.org/wiki/Binary_search voor meer informatie over de binary search.