Versie: 1.0

Door: Ben Kok - ©2020

Inhoudsopgave

PHP en MySQL

Algemeen

Notatiewijzen

camelCase, PascalCase, snake_case en kebab-case

Een array

```
$team = array("Ben","Iwan");
echo $team[1]; // geeft "Iwan"
```

Twee dimensionale array

```
$bedrijf = array(
  array("Ben","Iwan"),
  array("Diana","Eddie")
);
$bedrijf[1][1]; // geeft "Eddie"
```

Rekenkundige (Arithmethic) operatoren

** dit is een machtsverheffing (exponentiation) bv. $j**2 oftewel $j tot de macht 2

Toekennende operatoren

-= , min: $j -=3 is gelijk aan: $j = $j-3, zelfde geldt voor: *=, /=, .= en %= (daarnaast heb je uiteraard nog += en =)

Operands, expressies, operator associativity

- Het verschil tussen unary, binairy en tenary is het aantal **operands**, deze zijn resp. 1, 2 en 3.
- De meest simpele vormen van expressies zijn literals en variabelen, zij evalueren simpelweg zichzelf
- Operator associativity verwijst naar de richting van de verwerking deze kan van links naar rechts zijn of van rechts naar links.

Logische operatoren

xor is een exclusieve "or" vb: $j xor $x

xor is False zodra beide opties False zijn maar ook als beide opties True zijn! Dat laatste is het onderscheid t.o.v. de gewone "or"

Switch

```
switch $eenNaam{
        case:"Ben":
        echo "Hoi Ben";
        break;
case:"Enrico":
        echo "Hoi Chico";
        break;
default:
        echo "Hallo";
        break;
}
```

Je kunt de brackets in de switch ook vervangen door:

De begin bracket te vervangen door een :

En de eind bracket door "endswitch;"

Incrementing en decrementing van variabelen met de waarde 1:
$getal = 1;
++$getal = 2
--$getal = 1
Maar nu een "catch" zodra de ++ of -- aan de rechterkant staan gelden er iets andere regels.
Dit speelt een rol bij een conditie, dit wordt duidelijk in een voorbeeld:
$i = 4;
$i == $i++ deze geeft als waarde TRUE omdat EERST de vergelijking wordt uitgevoerd en daarna de variable $i met 1 wordt opgehoogd.
$i == ++$i geeft als waarde FALSE, het is simpelweg dus een kwestie van, van links naar rechts uitvoeren van de instructies.

Concatenation
Met de opdracht .= koppel je teksten aan elkaar.

Heredoc
```
echo <<<_LANGE_TEKST
```
Deze manier is voor een lange tekst geschikt zonder te gekke dingen, variabelen kunnen wel overigens en je kunt gewoon enteren etc.
De volgende regel geeft het einde aan van de heredoc en moet op een aparte regel staan.
```
_LANGE_TEKST;
```

Escaping van tekens
Met een string tussen dubbele aanhalingstekens kun je tekens escapen:
"Een teken zoals dit \t geeft een tab"

Echo tekst
Enters kunnen ook gewoon in de tekst opgenomen worden:
```
echo "Dit
   Werkt
      Ook gewoon in php";
```

Variabelen
Variabelen zijn "**loosely typed**" in PHP dat betekent dat ze niet gedeclareerd te worden voordat ze worden gebruikt.
Daarnaast zet PHP de variabele om in het type wat bij de context hoort.

Constanten

```
define("ROOT_LOCATION", "/usr/local/www");
```
Hierna kun je met **ROOT_LOCATION** de constante gebruiken.

Magische constanten
Deze constanten zijn een greep uit alle voor gedefinieerde constanten van PHP.
Bijzonder kenmerk zijn de twee underscores aan het begin en het einde, voorbeeld:
__FILE__ = het volledige pad en de bestandsnaam
__DIR__ = het volledige pad zonder de bestandsnaam, zelfde als dirname(__FILE);

Aandachtspunt: ELSEIF schrijf je aan elkaar in PHP
elseif = PHP
else if = Javascript

Echo en print hebben in PHP beide geen () nodig

Variabele Scope

Het beste uit te leggen met een voorbeeld

```
$temp = "De datum is:";
function lange_datum(){
	return $temp . " 15-08-1971";
}
```

$temp is buiten de functie gedeclareerd en kan in PHP niet in de functie worden bereikt.
Wat je kunt doen om dit op tel lossen is de functie uitbreiden:

```
function lange_datum($tekst){
	return $tekst . "15-08-1971";
}
```

En dan aanroepen met lange_datum($temp);

OF aangeven in de functie dat je de **global** variabele wilt gebruiken d.m.v.

```
function lange_datum(){
    global $temp;
    return $temp . " 15-08-1971";
}
```

Of je drukt de tekst eerder af en plaatst met de functie de datum

```
$temp = "De datum is:";
echo $temp . lange_datum();
function lange_datum(){
	return " 15-08-1971";
}
```

De waarde van een locale variabele stopt met bestaan wanneer de functie is uitgevoerd.

Static variabele
Hierbij wordt de waarde van de variabele bewaard in bijvoorbeeld de functie waarin deze is gedeclareerd. Binnen die functie kun je er bewerkingen op uitvoeren, iedere volgende aanroep van de functie kun je bijvoorbeeld de waarde verhogen van de static variabele. Static wordt gebruikt om een method zo te markeren dat deze bij de klasse hoort en niet bij het object

Superglobal variabelen

$GLOBALS = Alle variabelen die zijn gedefinieerd in de global scope van het script.
$_SERVER = Info over headers, paths en locaties van scripts.
$_GET, $_POST, $_FILES, $_COOKIES, $_SESSION, $_REQUEST (bevat $_GET, $_POST en $_COOKIE)
$_ENV = Variabelen doorgegeven aan het huidige script via de environment methode.

Expressions

Literal = letterlijk een getal of een tekst
2 = numerieke literal
'a' = string literal
FALSE = constante literal

Variabele = $e, $blabla etc..
Expression = een combinatie van waarden, variabelen, operators (+-/*etc) en functies die resulteren in een waarde.

TRUE of false, mag in zowel hoofd- als kleine letters. Kleine letters verdienen de voorkeur.
false print null en true print 1.

Loops

while

```
$x=1
while($x<10){
    echo('test' . $x)
    $x++;
}
```

do while

```
$x=0;
do{
    echo('halloo');
    $x++;
}while($x<10);
```

for

```
for($x=1,$y=20;$x < $y; $y--){
   echo("Xhallooo<br>");
}
```

Zodra er maar 1 uit te voeren regel is kun je het zonder de brackets doen, het hoeft niet, het is maar net wat de voorkeur heeft van jou of het team waarmee je samenwerkt.

Met het commando "**break**" kun je uit een loop breken.

```
for($x=1, $y=20;$x < $y; $y--){
   if($y==10) break;
   echo("Xhallooo<br>");
}
```

foreach loop

```
<?php
$colors = array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
  echo "$value <br>";
}
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $val) {
  echo "$x = $val<br>";
}
?>
```

Enkele aandachtspunten

- De **3 conditional statements** zijn if, switch en de ?: operator
- **De for loop is sterker dan de while** omdat de for loop twee additionele parameters ondersteunen die de loop kunnen controleren.

Array functies

Er zijn diverse Array functies binnen PHP hieronder enkele uitgelicht

is_array()	Hiermee kun je uitvinden of de variabele een array is.
count()	Telt de elementen van een array, met een extra optionele parameter, 0 is standaard en telt alleen het bovenste niveau, 1 telt ook eventuele onderliggende elementen van een multidimensionale array.
sort()	Om elementen te sorteren, sort() werkt op de bestaande array en geeft geen andere/nieuwe terug. Returnt TRUE bij succes en FALSE bij fout. Hier kun je ook een 2e optionele parameter meegeven zoals, SORT_NUMERIC of SORT_STRING
rsort()	Doet het omgekeerde van sort()
shuffle()	Om elementen van de array in een willekeurige volgorde te zetten
explode()	Om van een sting een array te maken. Je vervangt een teken, bijvoorbeeld de komma en alles voor en na de komma komt in een array als element.
extract() *	Hiermee kun je de key-value paren veranderen in variabelen, zie voorbeeld onder deze tabel. Het voorbeeld in het boek gaat over de $_GET en de $_POST variabelen, die beide associative arrays zijn.
compact()*	Doet eigenlijk het tegenovergestelde van extract(), deze maakt van allerlei variabelen een array. Zie voorbeeld onder de tabel.
reset()	Bij het doorlopen van bijvoorbeeld een foreach as loop, gaat de functie door de hele array heen m.b.v. een PHP pointer. Mocht het een keer nodig zijn om weer vanaf het begin de array te doorlopen, dan kun je met reset() weer opnieuw beginnen.
end()	end() werkt soortgelijk als reset() met dat verschil dat de pointer op het laatste element van de array wordt gezet.

*** extract() nadere uitleg**

```
$a = "Original";
$my_array = array("a" => "Cat","b" => "Dog", "c" => "Horse");
extract($my_array);
echo "\$a = $a; \$b = $b; \$c = $c";

Geeft=> $a = Cat; $b = Dog; $c = Horse
```

```
extract($_GET, EXTR_PREFIX_ALL, 'ben');
```

=> de nieuwe variabelen krijgen bij deze opdracht allemaal een extra prefix "ben_"
Deze manier verdient de voorkeur omdat anders niet helemaal duidelijk is welke variabelen er in gebruik zijn. In het andere voorbeeld hierboven zie je dat $a overschreven wordt.

*** compact() nadere uitleg**

```
$voornaam = "Ben";
$naam = "Kok";
$leeftijd  = "27";
```

$contact = compact($voornaam, $naam, $leeftijd);
en nu zitten ze in een array:

```
print_r($contact);
geeft=>
Array{
  [voornaam]=> Ben,
  [naam]=> Kok,
  [leeftijd]=> 27
}
```

Het commando "**continue**" is eigenlijk een "korte break", als de huidige iteratie aan de voorwaarde voldoet wordt deze overgeslagen en gaan we door met de volgende. Je breekt dus niet uit de loop maar uit de iteratie(s) die aan de voorwaarden voldoet/voldoen.

```
for($x=1, $y=20;$x < $y; $y--){
        if($y==15 || $y==16) continue;
        echo($y . "- Continue_Xhallooo 15<br>");
}
```

Implicit casting slaat op het loosely typed principe van PHP, dat variabelen dus automatisch gecast worden naar het type wat in de context hoort (of zou horen).

Implicit casting is niet altijd gewenst, gelukkig is er **explicit casting** wat inhoudt dat je ook zelf de types kunt casten zoals jij dat specifiek wilt. Enkel door bijv. (int) voor een variabele te zetten cast deze variabele naar een integer. Zo heb je ook nog (int), (string), (array), (bool), (float), (double), (real), (object), (integer) en (boolean) de laatste twee zijn gewoon voluit geschreven dezelfde als de verkorte versies.

Functies

Functies zijn case sensitive. De namen kunnen beginnen met letters of een underscore gevolgd door letters, cijfers of underscores.

Er zijn veel ingebouwde functies in PHP een paar voorbeelden voor Stringbewerking:

```
echo strrev("!einneB ollaH");
echo str_repeat(" - Hoi!", 9);
echo strtoupper(" hoera!!!");
```

PHP evalueert ieder element van binnenuit, ucfirst(strtolower("BLA bla BBBllllAAA"))
1. eerst wordt strtolower verwerkt => "bla bla bbbllllaaa"
2. en als laatste wordt ucfirst => "Bla bla bbllllaaa"

Functies parameters by reference

in plaats van strings rechtstreeks aan de functie door te geven, wijst u ze eerst toe aan variabelen en drukt u ze af om hun eerdere waarden te zien. Vervolgens roept u de functie aan zoals eerder, maar binnen de functiedefinitie plaatst u een & -symbool van elke parameter die door verwijzing moet worden doorgegeven.

De originele variabelen meegegeven aan de functie veranderen hiermee dus, een voorbeeld waardoor je het nooit vergeet:

```
//Pass een variabele by reference door een & teken voor het $ teken te zetten
$a1 = "WILLIAM";
$a2 = "henry";
$a3 = "gaTES";

echo $a1 . " " . $a2 . " " . $a3. "<br />";
fix_names($a1,$a2,$a3);
echo $a1 . " " . $a2 . " " . $a3. "<br />";

function fix_names(&$n1, &$n2, &$n3){ //HIER GEBEURT DE MAGIC!
        $n1 = ucfirst(strtolower($n1));
        $n2 = ucfirst(strtolower($n2));
        $n3 = ucfirst(strtolower($n3));
}
```

Versie conflicten PHP

Vanwege alle ingebouwde functionaliteit is het van belang de code robuust te maken. Je kunt testen of een functie bestaat door de functie "function_exists()" te gebruiken.

```
if (function_exists("array_combine")){
      echo("Function exists");
}else{
      echo("Function does not exists, build your own version");
}
```

Variabele scope

lokale variabelen, zijn alleen toegankelijk vanuit de code waar ze zijn gedefinieerd.
globale variabelen, zijn toegankelijk door de hele code heen
static variabalen, zijn binnen de functie toegankelijk en binnen de verschillende aanroepen houden ze hun waarde. Iedere aanroep kun je binnen de functie iets aan wijzigen aan de variabele.

Include files

Je kunt "include" gebruiken, maar beter nog is "**include_once**" om er zeker van te zijn dat er niet per ongeluk 2x dezelfde code wordt gebruikt.
Bij de functie "include" probeert PHP het bestand in te sluiten, maar het programma gaan gewoon verder als dat niet lukt.
Daarom is er een functie genaamd "require" met require wordt het bestand sowieso ge-include.
Ook hier wordt er weer aangeraden om "requiere_once" te gebruiken.

OOP en PHP

Het is een goede gewoonte om een klasse naam met een hoofdletter te beginnen.

```
class Klant{
    public $naam, $leeftijd;
    //Met de __construct functie kun je de parameters meegeven die in de haakjes staan bij New($parameter1,
$parameter2,[...])
    function __construct($naam, $leeftijd){
        $this->naam = $naam;
        $this->leeftijd = $leeftijd;
    }

    function get_Klantinfo(){
        $dagen = ($this->leeftijd *365);
        return $this->naam . " je bent grofweg " . $dagen . " dagen oud - get_Klantinfo()<br>";
    } //Je zou moeten kiezen in DIT geval of die hierboven of die hieronder te gebruiken
      //bij het creëren van een object wordt die hieronder immers altijd aangeroepen.
    function __destruct(){
        $dagen = ($this->leeftijd *365);
        echo $this->naam . " je bent grofweg " . $dagen . " dagen oud - __destruct<br>";
    }
}

$ben = new Klant("Ben",48);//Ben je bent grofweg 17520 dagen oud
$jessy = new Klant("Jessy",40);//Jessy je bent grofweg 14600 dagen oud
echo "<br>";
new Klant("Ralph",50);//Ralph je bent grofweg 18250 dagen oud
echo "<hr>";

$jan = new Klant("Klaas", 90);
echo $jan->naam . '<br>';
$jan->naam = "Benis";
echo $jan->naam . '<br>';
$jan->naam = "Klara";
echo $jan->naam . '<br>';


$jan->leeftijd = 43;
echo $jan->get_Klantinfo();//Klara je bent grofweg 15695 dagen oud
print_r($jan); //Klant Object ( [naam] => Johannes [leeftijd] => 43 )
```

Met de **__construct** functie kun je de parameters meegeven die in de haakjes staan bij New($parameter1, $parameter2,[...])

De **__destruct** function : Een destructor wordt aangeroepen wanneer het object wordt vernietigd of wanneer het script wordt gestopt of afgesloten. Oftewel aan het eind van het script wordt __destruct automatisch aangeroepen door PHP.

Een klasse kun je ook **impliciet** declareren bv:
class User{}
hierna kun je bijvoorbeeld toevoegen
$object1 = new User();
$object1 ->naam="Ben";
Dit is legitieme code, echter erg foutgevoelig en het geeft geen goed overzicht. Mocht er ergens een bug ontstaan, dan is deze door deze manier van programmeren lastig te achterhalen.

Properties en methoden

properties kun je oproepen vanuit het object op deze manier: $object->naam
(naam zonder $ er voor)
methoden kun je op deze manier oproepen: $object->eenmethode()

Met de **clone** operator kun je een nieuwe instantie maken van een klasse en creeer je een kopie zonder verwijzing naar het oorspronkelijke object.
$object2 = clone $object1

$this

De speciale variabele $this refereert altijd naar het huidige object waarbinnen de variabele zich bevind.

Enkele wetenswaardigheden over het **declareren van een variabelen met public:**

```
public $name = "Ben" //is goed
public $leeftijd = 25 //is goed zelfs beter omdat ik bijna het dubbele ben nu..
public $tijd = time() //is fout! omdat het een functie betreft
public $berekening = 1* $eengetal //is ook fout, omdat het een expressie bevat
```

In feite kun je **alleen** maar **literals** declareren op deze manier dus.

Constanten binnen een klasse

Op dezelfde manier hoe je met define een globale constante creëert, kun je binnen een klasse ook een globale constante creëren. Je kunt met **self::CONSTANTENAAM** aan de constante refereren

Voorbeeld:

```
Translate::lookup(); // met de klassenaam en :: kun je de methode aanroepen.
class Translate{
 const ENGELS = 0;
 const DUITS = 1;
 const NEDERLANDS = 2;
  static function lookup(){
    echo self::NEDERLANDS;
  }
}
```

Met self:: roep je direct de klasse aan, met $this een instantie van de klasse oftewel een object.

De eigenschap en methode scope

public = code buiten deze klasse heeft toegang tot de klasse en
extending klassen hebben ook overgeërfde toegang. Methoden krijgen automatisch de public property.
protected = code buiten deze klasse heeft geen toegang tot deze klasse maar
extending classes krijgen wel overgeërfde toegang
private = alleen toegang binnen de klasse zelf, en geen overgeërfde toegang voor de extending classes

static methods = een methode die wordt opgeroepen op een klasse maar niet op een object.
Een voorbeeld:

```
User::pwd_string();
Class User{
  static function pwd_string(){
    echo "Please enter your password";
  }
}
```

De dubbele :: wordt ook wel de **scope resolution operator** genoemd

static properties = een property die direct kan worden opgeroepen vanuit de klasse, er hoeft niet eerst een object van de klasse te worden aangemaakt.
Welk keyword wordt gebruikt om een method zo te markeren dat deze bij de klasse hoort en niet bij het object? **static**

```
<?php
$test = new Test();
$test->get_tekst();//Vanuit de methode met self:: "Een statische tekst"
echo("<hr>");
echo Test::$tekst;//"Een statische tekst";
echo("<hr>");
echo $test->tekst2;//"Nog een tekst"
echo("<hr>");
$test->get_tekst2();//"Nog een tekst@"

class Test{
        static $tekst = "Een statische tekst";
        public $tekst2 = "Nog een tekst";
        function get_tekst(){
                echo self::$tekst;
        }
        function get_tekstinhoud($tekst2){
                $this->tekst2 = $tekst2;
        }
        function get_tekst2(){
                echo $this->tekst2 . "@";
        }
}
?>
```

Gebruik maken van static doe je dus als je direct iets uit de klasse wilt halen (een property of een methode) en het scheelt heel wat code.

Vanuit een object/instance kan de static property niet opgeroepen worden:
$test = new Test();
$test->$tekst;//Dit geeft een foutmelding.
De methode get_tekst() werkt echter wel.

Overerving of Interitance

Je kunt van een superklasse dingen overerven door gebruik te maken in de subklasse van het keyword: extends

```
$ben = new Subscriber();
$ben->email ="b.kok@live.nl";
$ben->naam = "Ben";
$ben->telefoonnummer = "06";
$ben->display();//Ben06b.kok@live.nl
```

```
class User{
        public $naam, $telefoonnummer;
        function een_methode(){
                echo "een test xxx";
        }
}
class Subscriber extends User{
        public $email;
        function display(){
                echo $this->naam;
                echo $this->telefoonnummer;
                echo $this->email;
        }
}
```

Zodra je een zelfde naam voor een methode gebruikt in de subclass als in de parent class, dan overschrijft die in de subclass de parentclass.

Wil je vanuit de subclasse de parentclasse benaderen, dan kun je dat doen met het keyword **parent**

```
$aap = new Gorilla();
$aap->get_tekst();//"De gorilla subclass"
$aap->get_tekst_parent();//"De parent tekst"

class Aap{
        public $tekst = "De parent tekst";
        function get_tekst(){
                echo $this->tekst;
        }
}

class Gorilla extends Aap{
        public $tekst2 = "De gorilla subclass";
//zou je de variabele hierboven $tekst hebben genoemd dan was de output 2x "De gorilla subclass"
        function get_tekst(){
                echo $this->tekst2;
        }
        function get_tekst_parent(){
                echo parent::get_tekst();
        }
}
```

Als je er zeker van wilt zijn dat de code de methode van de huidige classe wordt opgeroepen dan kun je het self keyword gebruiken, bijvoorbeeld: self::method();

Subklasse constructors

Zodra je in een subclasse een eigen constructor wilt maken, moet je in die constructor altijd eerst de parent constuctor als eerste aanroepen. Daarna kun je jouw eigen code toevoegen.

```
$object = new Tiger();
echo "Tigers have...<br>";
echo "Fur: ". $object->fur ."<br>";
echo "Stripes:". $object->stripes;

class Wildcat{
        public $fur;
        function __construct(){
                $this->fur = "TRUE";
        }
}
class Tiger extends Wildcat{
        public $stripes;
        function __construct(){
                parent::__construct();
                $this->stripes = "TRUE";
        }
}
```

Final methods

WIl je voorkomen dat de subklassen een superklasse methode overschrijven gebruik dan de final methode:

```
class User{
  final function copyright(){
    echo("Niet aan mijn werk komen!");
  }
}
```

Enkele aandachtspunten:
Een functie kan standaard 1 waarde returnen, maar zodra je gebruikt maakt van arrays,references en globale variabelen kun je oneindig veel values returnen.

Zodra je een waarde van een variabele toekent aan een andere variabele dan is de waarde van de variabele gekopieerd. Maar zodra je variabele toekent aan de pointer van een variabele dan hou je een koppeling met de originele variabele en als je de waarde van de toegekende variabele veranderd dan veranderd ook de originele waarde.

De scope verwijst naar welke delen van het programma toegang heeft tot een variabele of methode.

Arrays

Een array maken kan op verschillende manieren:

```
<?php
$aap[]="Kiara";
$aap[]="Ben";
$aap[]="Enrico";
echo $aap[2];

//Of rechtstreeks met array:
$aap = array("Kiara", "Ben", "Enrico");
echo $aap[1];

Of met namen en een array:
$aap = array(
    naam1=>"Kiara",
    naam2=>"Ben",
    naam3=>"Enrico"
);
echo $aap=>naam1;
?>
```

Printf

Naast print() en echo hebben we ook nog **printf**. Deze functie is erg uitgebreid.
In de eerste parameter geef je met een % een pointer aan, en met een type letter bijvoorbeeld **s** aan dat het een string is. Dit kun je 1 of meer keer doen, voor al die aangegeven %pointers moet je een parameter geven op volgorde van links naar rechts. Dit kunnen variabelen, getallen en teksten zijn.

```
$x = 2-3;
$i = 20/2.3;
printf("%.2f of %d voorbeelden van %s printf",$i,$x,3); //8.70 of -1 voorbeelden van 3 printf
```

```
//Nog enkele voorbeelden van printf
echo("<pre>");
//15 posities
printf("Het resultaat is: %15f    \n", 123.42/ 12);
//15 posities gevuld met nullen 0
printf("Het resultaat is: %015f   \n", 123.42/ 12);
//15 posities 2 decimale precisie, rest met nullen opgevuld
printf("Het resultaat is: %015.2f   \n", 123.42/ 12);
//15 posities opgevuld met #, 2 dec precisie
printf("Het resultaat is: %'#15.2f   \n", 123.42/ 12);//let op de ' bij het hekje
echo("</pre>");
/*
Het resultaat is:       10.285000
Het resultaat is: 00000010.285000
Het resultaat is: 000000000010.29
Het resultaat is: ##########10.29
*/
```

De opbouw van een printf

% = pointer, '# geeft aan dat de lege ruimte met # wordt opgevuld,
  0 zonder ' zou de ruimte met 000...etc opvullen
15 is het totaal aantal posities
.2 is de precisie twee cijfers achter de komma
b,c,d,e, f (floating point), o, s (string), u, x (lowercase hex), X (uppercase hex)
En je kunt dus ook variabelen gebruiken, dat maakt deze functie ook heel interessant.

Voor strings is er nog een variatie

```
$naam = "Jan van der Doedel";
printf("Padding[ %s ]\n", $naam);//de gehele string
printf("Padding[ %-2.9s ]\n", $naam);//9 posities van de string vanaf links - 2 van rechts
printf("Padding[ %9.3s ]\n", $naam);//9 posities van de string vanaf links + 3 van rechts
printf("Padding[ %'#9.4s ]\n", $naam);//9 posities vanaf rechts 4 van de weergegeven string en de rest opgevuld met
#
//Het resultaat:
Padding[ Jan van der Doedel ]
Padding[ Jan van d ]
Padding[       Jan ]
Padding[ #####Jan  ]
```

Datum

```
printf(date('d-m-Y',time()));
echo'<hr>Weeknummer: ';
printf(date('W',time()));
echo'<hr>Schrikkeljaar: ';
printf(date('L',time()));
echo'<hr>Dag van de week: ';
$dagen=array('ma','di','wo','do','vr','za','zo');
printf(date('N',time()) . '<hr>Verkorte dagnaam: ' . $dagen[date('N',time())-1]);
echo'<hr>Dag van het jaar: ';
printf(date('z',time()));
echo'<hr>Uur:minuut:seconde: ';
printf(date('H:i:s',time()));
echo'--<hr>';
 function cal_days_in_year($year){
    $days=0;
    for($month=1;$month<=12;$month++){
        $days = $days + cal_days_in_month(CAL_GREGORIAN,$month,$year);
     }
 return $days;
}
echo cal_days_in_year(2020);
```

Het werken met files

Een voorbeeld:

```
$index2 = fopen("index2.php", "r");
$line = fgets($index2);
fclose($index2);
//echo $line;

echo('<br>***********<br>');

$index3 = fopen("index3.php", "r");
$text = fread($index3,30);
fclose($index3);
echo $text;

$delfile = "index3.php";
if(!unlink($delfile)){echo("Kon $delfile niet verwijderen!");}
else{echo("$delfile is verwijderd :)");}

//Een manier om een nieuwe file aan te maken
$mkphp = fopen("index3.php","w");
$tekst = <<<_END
<?php echo('TEST!!!'); ?>
_END;
    fwrite($mkphp, $tekst);
    fclose($mkphp);
echo('<br>---------------------------------<br>');


echo '</pre>';

echo('<br>***********<br>');

$index2 = fopen("index2.php", "r");
$line = fgets($index2);
fclose($index2);
//echo $line;

echo('<br>***********<br>');

$index3 = fopen("index3.php", "r");
$text = fread($index3,30);
fclose($index3);
echo $text;

//een bestand kopieren
copy("index.php", "index4.php");

//een bestand volledig lezen dit kan ook met een url
file_get_contents("tekst.txt");
```

De parameters

r	Lees file vanaf het begin	Alleen lezen, FALSE als bestand niet bestaat
r+	Lees file vanaf het begin en sta schrijven toe	Lezen en schrijven, FALSE als bestand niet bestaat
w	Schrijf file vanaf het begin en truncate file	Alleen schrijven, pointer aan het begin, truncate file tot 0 lengte. Als bestand niet bestaat creëer het.
w+	Schrijf file vanaf het begin en truncate file EN sta lezen toe	Lezen en schrijven, pointer aan het begin, truncate file tot 0 lengte. Als bestand niet bestaat creëer het.
a	Toevoegen aan het eind van het bestand	Alleen schrijven, pointer aan het eind van de file, indien file niet bestaat creëer het.
a+	Toevoegen aan het eind van het bestand EN sta lezen toe	Lezen en schrijven, pointer aan het eind van de file, indien file niet bestaat creëer het.

Uploaden files

```
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));
// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
  $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
  if($check !== false) {
    echo "File is an image - " . $check["mime"] . ".";
    $uploadOk = 1;
  } else {
    echo "File is not an image.";
    $uploadOk = 0;
  }
}
// Check if file already exists
if (file_exists($target_file)) {
  echo "Sorry, file already exists.";
  $uploadOk = 0;
}
// Check file size
if ($_FILES["fileToUpload"]["size"] > 500000) {
  echo "Sorry, your file is too large.";
  $uploadOk = 0;
}
// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
  echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
  $uploadOk = 0;
}
// Check if $uploadOk is set to 0 by an error
if ($uploadOk == 0) {
  echo "Sorry, your file was not uploaded.";
// if everything is ok, try to upload file
} else {
  if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file)) {
    echo "The file ". basename( $_FILES["fileToUpload"]["name"]). " has been uploaded.";
  } else {
    echo "Sorry, there was an error uploading your file.";
  }
}
```

Hierbij gaat het met name om de functie move_uploaded_file() en de globale variabele $_FILES die alle informatie bevat van de ge-uploadde bestanden.

Commando's geven voor het besturingssysteem vanuit php

```
$cmd = "whoami"; //Windows

exec(escapeshellcmd($cmd), $output, $status);

if ($status) echo "Exec command failed";
else
{
    echo "<pre>";
    foreach($output as $line) echo htmlspecialchars("$line\n");
    echo "</pre>";
}
```

HTML5 en XHTML

HTML5 is minder strikt,
 is prima terwijl het
 moet zijn in XHTML

MySQL

Fulltext

Zodra de FULLTEXT index is geactiveerd kan er met 'natuurlijke' taal gezocht worden op woorden 1 of meer. Daarnaast zijn er enkele 'stopwoorden' waarnaar niet gezocht wordt omdat ze vaak voorkomen op die manier wordt er met optimale snelheid gezocht. De lijst met stopwoorden kun je vinden op internet https://dev.mysql.com

Foreign Key toevoegen aan tabel

```
ALTER TABLE lid  ADD FOREIGN KEY (postcode) REFERENCES postcode(postcode);
```

Het vooraf verwijderen (DROP) van de Foreign keys is niet nodig wanneer de volgorde van verwijderen correct gebeurt bijvoorbeeld: eerst de email en telefoonnummer en dan het lid.

Daarnaast als FULLTEXT is toegepast op een aantal velden, dan kun je op een natuurlijke manier zoeken op die velden:

```
SELECT * FROM classics WHERE MATCH(author, title) AGAINST ('Kok +Ben -leven' IN BOOLEAN MODE);
```

Is de meest krachtige **MATCH AGAINST** query omdat je hierbij ook de + en - kunt gebruiken, dit komt door de toevoeging van IN BOOLEAN MODE;
=> Ben Kok | Van werkloos tot multimiljonair | Inspiratie | 202299977788831

```
SELECT * FROM classics WHERE MATCH(author, title) AGAINST ('Kok Ben leven');
```

Deze werkt ook, is ook heel krachtig en zoekt op alle woorden afzonderlijk

GROUP BY is geschikt voor bijvoorbeeld een COUNT omdat het geen rijen uitsluit. DISTINCT doet dat wel waardoor je rijen in de telling zou missen in dit voorbeeld.

```
CONCAT( 'bla','---', $variabele)
```

Met GROUP_CONCAT(DISTINCT [veld] SEPARATOR '
') aanvullende met GROUP BY [veld(en)] kun je een opsomming in 1 cel voor elkaar krijgen.

```
GROUP_CONCAT(DISTINCT emailadres SEPARATOR '<br>') As emailadres,
```

Limieten datatypen MySQL

CHAR data types		
CHAR (n)	Exact *n* (<=255)	CHAR(5) "Hallo" gebruikt 5 bytes CHAR(57) "Hallo" gebruikt 57 bytes
VARCHAR(n)	Tm *n* (<=65535)	VARCHAR(7) "Hallo" gebruikt 5 bytes VARCHAR(57) "Hallo" gebruikt 5 bytes
BINAIRE data types		
BINARY(n)	Exact *n* (<=255)	Zelfde als CHAR maar met binaire data
VARBINARY(n)	Tm *n* (<=65535)	Zelfde als VARCHAR maar met binaire data
TEXT data types		
TINYTEXT(n)	Tm *n* (<=255)	Zelfde behandeling als met een string met een karakterset
TEXT(n)	Tm *n* (<=65535)	Zelfde behandeling als met een string met een karakterset
MEDIUMTEXT(n)	Tm *n* (<=16.777.215)	Zelfde behandeling als met een string met een karakterset
LONGTEXT(n)	Tm *n* (<=4.294.967.295)	Zelfde behandeling als met een string met een karakterset
BLOB data types		
TINYBLOB(n)	Tm *n* (<=255)	Zelfde behandeling als binaire data
BLOB(n)	Tm *n* (<=65535)	Zelfde behandeling als binaire data
MEDIUMBLOB(n)	Tm *n* (<=16.777.215)	Zelfde behandeling als binaire data
LONGBLOB(n)	Tm *n* (<=4.294.967.295)	Zelfde behandeling als binaire data

Data type	Bytes used	Min waarde Signed	Min waarde Unsigned	Max waarde Signed	Max waarde Unsigned
TINYINT	1	-128	0	127	255
SMALLINT	2	-32.768	0	32.767	65.535
MEDIUMINT	3	-8.388.608 min 8 miljoen	0	8.388.607 8 miljoen	16.777.215 16 miljoen
INT/INTEGER	4	-2.147.483.648 min -2,14 miljard	0	2.147.483.648 2,14 miljard	4.294.967.295 4,29 miljard
BIGINT	8	-9.223.372.036.854.775.808	0	9.223.372.036.854.775.807	18.446.744.073.709.551.615
FLOAT	4	-3.40 met 38 nullen	Niet aw.	3.40 met 38 nullen	Niet aw.
DOUBLE/REAL	8	-1.80 met 308 nullen	Niet aw.	1.80 met 308 nullen	Niet aw.

Transactions

In sommige applicaties is de volgorde van uitvoering van de query's belangrijk, bijvoorbeeld bij het storten van geld, je wilt dat het eerst van de rekening af gaat en daarna gestort wordt. Zodra het bijvoorbeeld niet gestort wordt of niet afgeschreven is er een probleem. Pas als beide gedaan zijn is het goed, hier komt de "transaction" goed van pas.

De transacties in MySql starten met "**Begin**" of "**START TRANSACTION**"

Zodra je tevreden bent met een serie van query's die succesvol zijn uitgevoerd dan kun je de "**COMMIT**" uitvoeren. Deze zorgt ervoor dat al de query's worden uitgevoerd in de database. Totdat het COMMIT commando wordt gegeven gaat MySQL er vanuit dat de query's tijdelijk zijn. Dit geeft de mogelijkheid een **ROLLBACK** commando te geven waardoor alles terug wordt gedraaid naar de situatie bij de start van de transaction.

Voorbeelden:

```
BEGIN;
UPDATE accounts SET name='Ben' WHERE id=1;
UPDATE accounts SET name='Enrico' WHERE id=2;
SELECT * FROM accounts;
```

Twee opties:
Zolang er nog geen COMMIT is geweest kun je de query met ROLLBACK terugdraaien
Check met SELECT * FROM accounts of het is gelukt.

Of

COMMIT; //wijzigingen worden vastgelegd
als er tussendoor iets fout gaat zal de COMMIT niet worden uitgevoerd en worden er geen wijzigingen in de database gemaakt worden.

EXPLAIN
Het commando EXPLAIN geeft meer informatie over de query die je maakt, deze informatie kun je gebruiken om een betere/ efficiëntere query te maken.

```
Een backup file maken van de database
mysqldump -u username -p password databasenaam > databasenaam.sql;

Een backup inlezen vanuit de commandline
mysql -u user -ppassword -D databasenaam < databasenaam.sql;
```

Normaliseren

- **Nulde** normaalvorm bevat alle data
- **Eerste** normaalvorm: kolommen die dezelfde soort data bevatten naar een andere tabel geplaatst
- **Tweede** normaalvorm.
  data die redundant is over meerdere rijen naar een andere tabel verplaatst
- **Derde** normaalvorm:
  normaalvorm wordt data die niet direct afhankelijk is van de primaire sleutel, maar van een andere sleutel in een andere tabel geplaatst

Nulde normaalvorm

Kolommen verwijderen die berekend of afgeleid kunnen worden van andere velden

Eerste normaalvorm

1. Alle **kolommen** met exact dezelfde gegevens moeten verwijderd worden
2. Alle **kolommen** moeten 1 waarde bevatten per stuk
3. Er moet een **primaire sleutel** zijn die **uniek** is voor iedere rij autoincremet

Redundante data verspreid over meerdere rijen wordt in een aparte tabel geplaatst.
In de eerste normaalvorm worden kolommen met dezelfde soort data in een andere tabel geplaatst.

Tweede normaalvorm

Gegevens afsplitsen van maar een deel van de primary key, stel er zijn 3 velden die primary key zijn en er is data verbonden met 1 key dan kun je die groeperen in een nieuwe tabel.
De data die alleen maar is verbonden met alledrie de keys horen gewoon thuis in de oorspronkelijke tabel.

Derde normaalvorm

Data die niet direct afhankelijk is van de primaire sleutel, maar van een andere sleutel wordt in een andere tabel geplaatst.
Gegevens afsplitsen die afhankelijk zijn van een non-key waarde

Voorbeelden

Factuurnr	Factuurdatum	Naam	Klantnr	Adres	Postcode	Plaats	Productcode
Productdatum	Productomschrijving	Productaantal	Productprijs	Totaalbedrag			
0e Normaalvorm							
Factuurnr	Factuurdatum	Naam	Klantnr	Adres	Postcode	Plaats	Productcode
Productdatum	Productomschrijving	Productaantal	Productprijs	Totaalbedrag			
Totaalbedrag	kan berekend worden uit Productaantal en Productprijs						
1e Normaalvorm							
Tabel							
Factuur	**Factuurnr**	Factuurdatum	Naam	Klantnr	Adres	Postcode	Plaats
Factuurregel	**Factuurnr**	**Productcode**	**Productdatum**	Productomschrijving	Productaantal	Productprijs	
1)Primary key nodig en 2) afsplitsen van herhalende gegevens/groepen							
2e Normaalvorm							
Tabel							
Factuur	**Factuurnr**	Factuurdatum	Naam	Klantnr	Adres	Postcode	Plaats
Factuurregel	**Factuurnr**	**Productcode**	**Productdatum**	Productaantal			
Product	**Productcode**	Productomschrijving	Productprijs				
1) Gegevens afsplitsen die afhankelijk zijn van maar een deel van de primary key							
3e Normaalvorm							
Tabel							
Factuur	**Factuurnr**	Factuurdatum	Klantnr				
Factuurregel	**Factuurnr**	**Productcode**	**Productdatum**	Productaantal			
Product	**Productcode**	Productomschrijving	Productprijs				
Klant	**Klantnr**	Naam	Adres	Postcode	Plaats		
1) Gegevens afsplitsen die afhankelijk zijn van een non-key waarde							

PHP connectie naar MySQL

$_POST en $_GET zijn geassocieerde arrays.

Verbinding met MySQL en weergave van data

$row = $result->fetch_array(**MYSQLI_ASSOC**);//**MYSQLI_NUM** is ook nog een optie

Sluiten van de verbinding

$result->close();// alleen bij SELECT als er een resultaat doorlopen is
$conn->close();// deze moet altijd als $conn bestaat

AUTO_INCREMENT, het laatst ingevoerde ID achterhalen

Het is soms handig het ID te weten wat dmv AUTO_INCREMENT is ingevoerd.
Dit kan met de mysql_insert_id function en vraag je op als volgt:
$conn->insert_id; je zou de waarde in een variabele kunnen stoppen zodat je deze bij voorbeeld in een daarop volgende query kunt gebruiken.

Subquery

```
//Een subquery kan ook op deze manier binnen een parent for lus
$subquery = "SELECT * FROM andere_tabel WHERE name='" . $row[$k]."';";
$subresult= $conn->query($subquery);
//...etcetera
```

Beveiliging

```
function get_post($conn, $var){
    return $conn->real_escape_string($_POST[$var]);
}
```

Doormiddel van de real_escape_string die vanuit het $conn object kan worden gebruikt, worden alle quotes ge-escaped. Dit is nodig omdat hackers vaak als eerste met quotes gaan testen of ze toegang kunnen krijgen tot een database.
Vroeger werd magic_quotes gebruikt, maar deze is inmiddels deprecated.

```
function sanitizeString($var){
    //Check of magic quotes worden gebruikt en zo ja verwijder de toegevoegde slashes
    if(get_magic_quotes_gpc())
        $var = stripslashes($var);
    //Haal alle HTML slashes weg
    $var = strip_tags($var);
    //Zet alle overige tags of tekens om in onschadelijke karakters als < naar-> &gt;
    $var = htmlentities($var);
    return $var;
}

function sanitizeSQL($conn, $var){
    //De real_escape_string houdt rekening met de huidige karakterset die wordt gebruikt
    //vandaar dat deze gebruikt moet worden vanuit het mysqli connectie object
    $var = $conn->real_escape_string($var);
    $var = satitizeString($var);
    return $var;
}
```

Placeholders!

In MySQL is het vrij omslachtig:
Als eerste bereid je het statement voor:

```
PREPARE statement FROM "INSERT INTO classics VALUES(?,?,?,?)";
SET @author = $_POST['author'],
        @title = $_POST['title'], ....;
EXECUTE statement USING @author, @title, ...etc..;
DEALLOCATE PREPARE statement;
```

Maar in PHP maakt de mysqli extension het makkelijker:

```
//Connectie opbouwen en dan
$author = $_POST['author'];
//Etc...
$stmt = $conn->prepare('INSERT INTO classics VALUES(?,?,?,?,?);');
$stmt->bind_param('ssss', $author, $title, $category, $isbn);
//De eerste parameter geeft aan van welk type de velden zijn
$stmt->execute();
printf("%d row(s) inserted.\n", $stmt->affected_rows);
$stmt->close();
$conn->close();
```

Om het datatype aan te geven gebruik je de volgende letters:
i = integer, d = double, s = string, b = blob, deze wordt in pakketjes verstuurd

Beveiligen tegen een XSS attack

Gebruik voor de alle mogelijke invoer mogelijkheden, inputs, textarea etc.. de htmlentities() functie, deze stript de HTML markup en laat de tags zien maar deze wordt niet uitgevoerd het is de enkel weergave van de ingevoerde malicious code. Gecombineerd met de vorige functie die MySQL attacks voorkomt, hier de code die ook XSS voorkomt:

```
$password = mysql_entities_fix_string($conn, $_POST['user']);
function mysql_entities_fix_string($conn, $string){
    return htmlentities(mysql_fix_string($conn, $string));
}
function mysql_fix_string($conn, $string){
    if (get_magic_quotes_gpc()) $string = stripslashes($string);
    return $conn->real_escape_string($string);
}
```

Procedurele manier van mysqli, een voorbeeld hieronder:

```
$link = mysqli_connect($hn, $un, $pw, $db);
if(mysql_connect_errno()) die("Fatal error");
$result = mysqli_query($link, "SELECT * FROM classics");
$rows = msql_num_rows($result);
//De numerieke array doorlopen en het resultaat weergeven
$row = mysqli_fetch_array($result, MYSQLI_NUM);
//Bijvoorbeeld $row[0]; kun je weergeven
//En wil je de id weten van een pas ingevoerd record dan doe je dat zo:
$insertID = mysqli_insert_id($result);
//Om strings te escapen
$escaped = mysqli_real_escape_string($link, $string);
//Preparen van een query:
$stmt = mysql_prepare($link, 'INSERT INTO classic VALUES(?,?,?,?)');
//Om de variabelen te binden:
mysql_stmt_bind_param($stmt, 'ssss', $author, $title, $category, $isbn);
//Execute query
mysqli_stmt_execute($stmt);
//Close objects
mysqli_stmt_close($stmt);
mysqli_close($link);
```

Een aandachtspunt bij MySQL is dat zodra er dependenties zijn (FK), het van belang is dat queries als updaten, deleten of insert op **de juiste volgorde** worden afgehandeld. Het werken met FK voorkomt dat er wezen zijn en dwingt een bepaalde werkwijze af.

Cookies, Sessions en Authenticatie

Cookies kunnen alfanumerieke gegevens bevatten tot een grootte tot 4 Kb.
Die gegevens kunnen uitgewisseld worden tussen jouw computer en de server.

Een cookie setten

```
setcookie(name, value, expire, path, domain, secure, httponly);
```

Uitleg van de parameters:
name, verplicht, de naam van het cookie
value, verplicht, de waarde van het cookie, of de inhoud, deze kan oplopen tot 4Kb
expire, optioneel, de Unix timestamp van de expiratiedatum. Vaak gebruik je hier time() + een aantal seconden. Indien er geen waarde wordt meegegeven dan verdwijnen de cookies zodra de browser wordt gesloten.
path, optioneel, het pad van het cookie op de server. Een forward slash geeft aan dat deze bereik heeft over het gehele domein. Zodra er een subdirectory in staat heeft de cookie alleen daarop betrekking.
domain, optioneel, Dit is het internet domein of subdomein van het cookie. Is het de root van het domein dan geldt het ook voor alle subdomeinen 'benkok.nl' geldt dan ook voor 'www.benkok.nl' en 'tools.benkok.nl' etc..
secure, optioneel, als het cookie vanuit een https omgeving komt is de waarde TRUE anders FALSE.
httponly, optioneel, http protocol afdwingen op die manier kunnen scripting talen niet bij het cookie. Default staat deze waarde op FALSE

Voorbeeld van een cookie met de naam location en de waarde USA, expired over een week, beschikbaar over het gehele domein:

```
setcookie('location', 'USA', time() + 60 * 60 * 24 *7, '/');
```

Toegang krijgen tot de gegevens van een cookie:

```
if(isset($_COOKIE['location'])) $location = $_COOKIE['location'];
```

Verwijderen/ vernietigen van een cookie

```
setcookie('location', 'USA', time() -2592000, '/');
```

Aandachtspunten:

- Cookies worden verstuurd voordat de eigenlijke HTML-code wordt verstuurd.
- PHP kan prima overweg met third party cookies.

HTTP Authenticatie

Deze optie moet wel ingeschakeld staan op de server, maar dat is meestal zo.

```
if( isset($_SERVER['PHP_AUTH_USER']) && isset($_SERVER['PHP_AUTH_PW']) ){
    $un_temp = sanitizeSQL($conn, $_SERVER['PHP_AUTH_USER']);
    $pw_temp = sanitizeSQL($conn, $_SERVER['PHP_AUTH_PW']);
    $query   = "SELECT * FROM users WHERE username='$un_temp'";
    $result  = $conn->query($query);
    if(!$result) die('User not found');
    elseif ($result->num_rows){
        $row = $result->fetch_array(MYSQLI_NUM);
        $result->close();

        if( password_verify($pw_temp, $row[3])){
            echo sanitizeString("$row[0] $row[1]:
            Hi $row[0], you are now logged in as '$row[2]'");
        }else{
            die('Invalid user/password combination.');
        }
    }else{
        die('Invalid user/password combination.');
    }

}else{
    header('WWW-Authenticate: Basic realm="Restricted Area"');
    header('HTTP/1.0 401 Unauthorized');
    die("Please enter your username and password");
}
```

'WWW-Authenticate: Basic realm="**Restricted Area**"
"Restricted Area", is de naam van de beschermde omgeving, deze mag je zelf bedenken/bepalen.

Het is overigens wel zo dat deze manier van authenticatie wordt afgeraden zodra er veel gebruikers inloggen op de webserver, het kost namelijk redelijk wat rekenkracht. Beter is dan om sessies te gebruiken voor de authenticatie.

Bij het vergelijken van gebruikersnamen en wachtwoorden wordt het === (de identity) operator gebruikt en niet de ==(is gelijk aan) operator. Dit is om nodig om een exacte vergelijking te maken.

Gebruikersnamen en wachtwoorden

Om te voorkomen dat we de wachtwoorden als 'clear text' op slaan maken we gebruik van de zogenaamde 'one-way function'. Dit is een simpele functie die het wachtwoord omzet in een willekeurige string.

De password_hash

Meteen even een voorbeeld van deze functie met daarna de uitleg:

```
echo password_hash('mijnwachtwoord', PASSWORD_DEFAULT);
```

De laatste parameter PASSWORD_DEFAULT, gebruikt altijd de meest actuele en veilige hash methode die er op dit moment beschikbaar is. Daarnaast genereert password_hash telkens een willekeurige salt voor ieder wachtwoord. Het zelf toevoegen van een eigen 'salt' wordt afgeraden vanwege security risico's.
Denk er om dat er in de database een veld van tenminste 255 karakters moet worden gecreëerd voor de wachtwoord string met de hashes. Er zijn ook andere parameters in te stellen in plaats van PASSWORD_DEFAULT, maar de aanbeveling is om **PASSWORD_DEFAULT** aan te houden.

Om het wachtwoord te verifieren gebruik je password_verify()

```
if(password_verify("mijnwachtwoord", $hash))
        echo "valid";
```

Sessions

Om er zeker van te zijn dat de sessie variabelen worden gebruikt door de juiste gebruiker, bewaart PHP een cookie in de webbrowser van de gebruiker. Op die manier kan PHP de gebruiker uniek identificeren.
In die zin moeten we altijd een check op cookies uitvoeren, als er geen cookies zijn dan ook geen toegang tot de applicatie.
Een sessie starten is vrij simpel, dat doe je door te beginnen met de functie **session_start()**
Een sessie variabele maken doe je bijvoorbeeld als volgt: $_SESSION['naam'] = "Een naam";

Om een sessie uit te loggen kan het volgende script gebruikt worden, het ruimt alles op en logt de gebruiker uit.

```
function destroy_session_and_data(){
    session_start();
    $_SESSION = array();
    setcookie(session_name(), '', time() - 2592000, '/');
    session_destroy();
}
```

Altijd een timeout instellen voor alle veiligheid

Sommige gebruikers vergeten wel eens uit te loggen, waardoor er een veiligheidslek ontstaat.
Daarom is een timeout instellen een goede gewoonte, in PHP doe je dat bijvoorbeeld op de volgende manier:
ini_set('session.gc_maxlifetime', 60 * 60 * 24); //In dit geval is de timeout op een dag vastgesteld
Als je wilt weten wat de huidige timeout is van een sessie, gebruik dan deze code:
echo ini_get('session.gc_maxlifetime');

Session Hijacking

De beste manier om session hijacking te voorkomen is door gebruik te maken van TLS.
Hieronder een voorbeeld van code die ook bescherming biedt als TLS niet een optie is.

```
$_SESSION['check'] = hash('ripemd128', $_SERVER['REMOTE_ADDR'] . $_SERVER['HTTP_USER_AGENT']);
```

Deze code kun je verwerken in authenticate.php
En onderstaande code in de pagina's die de beveiliging nodig hebben:

```
function check_session(){
    if($_SESSION['check'] != hash('ripemd128', $_SERVER['REMOTE_ADDR'] . $_SERVER['HTTP_USER_AGENT']))
different_user();
//  if($_SESSION['check'] != 'X'. hash('ripemd128', $_SERVER['REMOTE_ADDR'] . $_SERVER['HTTP_USER_AGENT']))
different_user();
}
```

Met de regel in commentaar kun je testscenario's uitvoeren.

Session Fixation

Zodra iemand met kwaadaardige bedoelingen een valide sessieID verkrijgt, bijvoorbeeld door een server gegenereerd, en deze door de nietsvermoedende gebruiker laat gebruiken om mee in te loggen ontstaat er een lek waardoor de hacker zichzelf toegang kan verschaffen tot diegene zijn gegevens. Dit wordt makkelijk gemaakt zodra er gewerkt wordt met de sessie_id in de url, bijvoorbeeld: http://localhost/index.php?sessie=292929293

Om dit te voorkomen maken we gebruik van de functie session_regenerate_id();

Dit gebruik je als bijvoorbeeld hieronder en bij voorkeur natuurlijk helemaal bovenaan in de code.

```
session_start();
if(!isset($_SESSION['initiated'])){
     session_regenerate_id();
     $_SESSION['initiated'] = 1;
}
```

Forceren van alleen cookie-only sessies

Dit doe je simpelweg met door deze code toe te voegen:

```
ini_set('session.use_only_cookies', 1);
```

Uiteraard de gebruikers of bezoekers van respectievelijk de applicatie of website wel even informeren dat "cookies" aan moeten staan.

Een shared server

Op een shared server loop je de kans dat al de sessies bewaard worden op dezelfde plaats en op die manier is dat een veiligheidsrisico. Om dat te voorkomen geef je het pad mee in de volgende functie.

```
ini_set('session.save_path', '/home/user/myaccount/sessions/');
```

Deze session map zal snel groeien, daarom moet deze aldoor goed worden opgeruimd

Nog enkele tips
PHP
Constanten definieren vb:
Define('MYSQL_SERVER','localhost');
Waarom gebruiken? Constanten gedragen zich ook als globale variabelen die je ook kunt gebruiken in bv. Functies. Bovendien Compressie gebruiken
Snelheid laden van webpagina optimaliseren
@ob_start('ob_gzhandler') aanroepen in het begin van een webpagina

Beveiligen
Plaats de config bestanden met wachtwoorden in een directory die niet te benaderen is vanaf het www
Server side includes
Gebruik include en include_once voor statische html inhoud en
Gebruik require en require_once voor functies e.d. , reden is dat require eerst geparst wordt

Voorbeeld van een config_inc.php voor lokaal testen en online

```
if ($_SERVER['SERVER_ADDR'] == '127.0.0.1') {
   // Lokale MySQL-databaseserver
   define('MYSQL_SERVER',        'localhost');
   define('MYSQL_GEBRUIKERSNAAM', 'root');
   define('MYSQL_WACHTWOORD',     '');
   define('MYSQL_DATABASENAAM',   'test');
   // Lokaal alle foutmeldingen weergeven
   error_reporting(E_ALL | E_STRICT);
} else {
   // Externe MySQL-databaseserver
   define('MYSQL_SERVER',        'INVULLEN');
   define('MYSQL_GEBRUIKERSNAAM', 'INVULLEN');
   define('MYSQL_WACHTWOORD',     'INVULLEN');
   define('MYSQL_DATABASENAAM',   'INVULLEN');
   // Alle foutmeldingen op de live server uitschakelen
   error_reporting(0);
}
// De locale instellen op Nederlands en Nederland en de datum- en tijdzone instellen op Amsterdam.
setlocale(LC_ALL, 'nl_NL', 'nld_nld', 'Dutch_Netherlands');
date_default_timezone_set('Europe/Amsterdam');
@ini_set('date.timezone', 'Europe/Amsterdam');
?>
```

Tenary operator
Een voorbeeld van een tenary operator (if then else in 1 regel):

```
"Hallo" . ($gebruikersnaam == "Ben")?"Bennie":$gebruikersnaam;
```

Verbinding opzetten naar MySQL met Mysqli

```
$mysqli = new mysqli("localhost", $config_dbuser, $config_dbpass, $config_dbname);
if ($mysqli->connect_errno) die ("De database is niet bereikbaar.".$mysqli->connect_errno);
if (!$mysqli->set_charset('utf8')) die("Character set kan niet worden geladen.");
$result = $mysqli->query("SELECT naam FROM T_tabel ");
if ($row = $result->fetch_assoc()){
    $_SESSION["naam"] = $row["naam"];
}
```

```
$result->free();
$mysqli->close();
?>
```

Variabele variabele $$

```
$user = “John”;
$$user = “Man”;
```

echo $John –> geeft als resultaat: “*Man*“;

Array

Met de functie array(‘a’,’b’,’c’);

is hetzelfde als arary(1=>’a’,2=>’b’,3=>’c’);

arrays kun je ook multi dimensionaal maken, een array in een array

```
$arr = array(1=> array(“naam” => “Ben”,”leeftijd”=>28),array(“naam”=>”Enrico”,”leeftijd”=>9));
$arr[1][“naam”] = “Ben”
```

Met een foreach loop kun je een array doorlopen, vb:

```
$spelers = array(“Jan”,”Piet”,”Klaas”);
foreach ($spelers as $key => $value){
        print(“#$key = $value “);
}
```

output:

#0 = “Jan”

#1 = “Piet”

#3 = “Klaas”

Cast type

```
$str = “1”;
$nummer = (int) $str;
```

Zo heb je ook: (float), (real), (double), (string), (bool), (array), (object)

MySQL

MYSQL procedure

```
CREATE PROCEDURE Adres()
  BEGIN
       SELECT  postcode bla bla, query  die je normaal bij een view schrijft
   END
```

Vervolgens hoef je alleen nog maar de procedure aan te roepen in MYSQL in dit geval: CALL Adres()

RLIKE in MySql

Met RLIKE kun je in 1 statement op meerdere values zoeken

```
SELECT naam, telefoon FROM T_tabel WHERE telefoon RLIKE (050|020|030)
```

RLIKE maakt gebruik van regular expressions, vandaar de R.

Grote bestanden uploaden

In php.ini de volgende parameters aanpassen in bijvoorbeeld:

```
post_max_size=500M
upload_max_filesize=500M
memory_limit=900M
```