

Webdesign 4

Gabriel Sánchez Cano



Meer informatie over deze en andere uitgaven kunt u verkrijgen bij:

BIM Media B.V.
Postbus 16262
2500 BG Den Haag
Tel.: (070) 304 67 77
www.bimmedia.nl

Gebruik onderstaande code om dit boek eenmalig toe te voegen aan je boekenplank op www.academicx.nl.
Let op: je kunt deze code maar één keer gebruiken.

© 2014 BIM Media B.V., Den Haag
Academic Service is een imprint van BIM Media B.V.

1e druk 2007
2e druk 2010
3e druk 2012
4e druk 2014

Vormgeving: Studio Bassa, Culemborg
Zetwerk: Redactiebureau Ron Heijer, Markelo

ISBN: 978 90 395 2781 8
NUR: 124

Alle rechten voorbehouden. Alle auteursrechten en databankrechten ten aanzien van deze uitgave worden uitdrukkelijk voorbehouden. Deze rechten berusten bij BIM Media B.V.

Behoudens de in of krachtens de Auteurswet gestelde uitzonderingen, mag niets uit deze uitgave worden verveelvoudigd, opgeslagen in een geautomatiseerd gegevensbestand of openbaar gemaakt in enige vorm of op enige wijze, hetzij elektro-nisch, mechanisch, door fotokopieën, opnemen of enige andere manier, zonder voorafgaande schriftelijke toestemming van de uitgever.

Voorzover het maken van reprografische verveelvoudigingen uit deze uitgave is toegestaan op grond van artikel 16 h Auteurswet, dient men de daarvoor wettelijk verschuldigde vergoedingen te voldoen aan de Stichting Reprorecht (postbus 3060, 2130 KB Hoofddorp, www.reprorecht.nl). Voor het overnemen van gedeelte(n) uit deze uitgave in bloemlezingen, readers en andere compilatiswerken (artikel 16 Auteurswet) dient men zich te wenden tot de Stichting PRO (Stichting Publicatie- en Reproductierechten Organisatie, Postbus 3060, 2130 KB Hoofddorp, www.cedar.nl/pro). Voor het overnemen van een gedeelte van deze uitgave ten behoeve van commerciële doeleinden dient men zich te wenden tot de uitgever. Hoewel aan de totstandkoming van deze uitgave de uiterste zorg is besteed, kan voor de afwezigheid van eventuele (druk) fouten en onvolledigheden niet worden ingestaan en aanvaarden de auteur(s), redacteur(en) en uitgever deswege geen aansprakelijkheid voor de gevolgen van eventueel voorkomende fouten en onvolledigheden.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the publisher's prior consent. While every effort has been made to ensure the reliability of the information presented in this publication, BIM Media B.V. neither guarantees the accuracy of the data contained herein nor accepts responsibility for errors or omissions or their consequences.

Voorwoord

Webdesign 4 is bestemd voor alle opleidingen MBO niveau 4 waarbij het ontwikkelen van webomgevingen een rol speelt, waaronder Applicatie- en mediaontwikkeling, Mediavormgever en Mediatechnologie. Het is geschreven op basis van de nieuwe competentiegerichte kwalificatiestructuur van ECABO en is daarom uitstekend geschikt voor zelfstudie. Alle hoofdstukken zijn voorzien van voldoende praktijkgerichte opgaven waarmee de theorie direct in praktijk kan worden gebracht. Daarmee is dit boek ook een praktische inleiding voor iedereen die zelf een website wil bouwen.

Dit boek is geschreven met de recente ontwikkelingen van het web in gedachten. Web 2.0 is de tweede generatie van op internet gebaseerde services, zoals sociale netwerken en wiki's (Wikipedia). Bij Web 2.0 ligt de nadruk op samenwerking en het delen van content tussen de gebruikers. De mogelijkheden die Web 2.0 en de nieuwe open-source technologieën voor de web-developers bieden zijn zeer interessant en komen daarom uitgebreid aan de orde.

In deze vierde druk is het hoofdstuk over JavaScript geheel herzien en aanzienlijk uitgebreid. Sinds de derde druk zijn de eerste twee hoofdstukken gebaseerd op HTML5 en CSS3, en bevat het hoofdstuk over UML een paragraaf over klassendiagrammen. Een oefentoets en grotere praktijkopdrachten zijn beschikbaar via de portal www.PraxticX.nl. Voor docenten die het boek voorschrijven zijn grote portfolio-opdrachten beschikbaar.

Ik wil al mijn studenten en collegae van het ROC van Amsterdam bedanken voor hun feedback tijdens het maken van dit boek. Speciale dank voor Louis Trautwein voor zijn zorgvuldige commentaar. Daarnaast bedank ik Paul Post van Academic Service voor de plezierige en deskundige samenwerking. Ik heb dit boek met veel plezier geschreven en ik hoop dat zowel studenten als docenten er met veel plezier mee zullen werken.

Amsterdam, maart 2014
Gabriel Sánchez Cano



Inhoud

1	HTML5 en CSS3	1
1.1	Inleiding HTML5	1
1.2	Meta-informatie	3
1.3	Pagina-markup	6
1.4	Navigatie markup	8
1.5	Inleiding CSS3	10
1.6	Navigatiedesign	11
1.7	Logo markup	21
1.8	Logo design	22
1.9	Article markup	25
1.10	Article design	27
1.11	Aside markup	30
1.12	Aside design	33
1.13	Footer markup	36
1.14	Footer design	37
2	Rich Media Sites	41
2.1	Markup van de pagina Inschrijven	41
2.2	Design van de pagina Inschrijven	47
2.3	Markup van de pagina Nieuwsbrief	53
2.4	Design van de pagina Nieuwsbrief	55
2.5	Markup van de Resultatenpagina	55
2.6	Design van de Resultatenpagina	60
3	JavaScript	63
3.1	Inleiding JavaScript	63
3.2	De scriptingtaal JavaScript	66
3.3	Ingebouwde functies	66
3.4	JavaScript-variabelen	68
3.5	Datatypes	71
3.6	Het Array-object	76
3.7	Array-methodes	80
3.8	Hash-arrays	89
3.9	Het Date-object	93
3.10	Beslissingsstructuren	97
3.11	Eigen functies	108
3.12	Lussen	122
3.13	Het Math-object	135
3.14	String-methodes	138
3.15	Algoritmes	144

3.16	Het Document Object Model	148
3.17	Cookies	160
3.18	Een winkelwagentje in JavaScript	165
4	PHP	171
4.1	Inleiding PHP	171
4.2	De PHP-programmeertaal	173
4.3	PHP-variabelen	175
4.4	Arrays	182
4.5	De if-opdracht	185
4.6	Variabelen uit een formulier	188
4.7	Switch	193
4.8	Functies	196
4.9	Externe functies	200
4.10	De for-lus	203
4.11	De while-lus	206
4.12	De foreach-lus	210
4.13	Constanten en globale variabelen	215
4.14	De functie date()	219
4.15	Cookies?	221
4.16	Reguliere expressies	223
4.17	String functies	228
4.18	Gegevensbestanden	232
5	Databases	239
5.1	Inleiding databases	239
5.2	Normaliseren	241
5.3	Normalisatieproces	242
5.4	Data modellering	250
6	SQL	255
6.1	Wat is SQL?	255
6.2	MySQL	255
6.3	De databaseadministrator	260
6.4	MySQL stringfuncties	275
6.5	Stored programma's in MySQL-databases	283
6.6	Programmering met PHP en MySQL	294
7	UML	305
7.1	Inleiding UML	305
7.2	Modelleren met UML	307
7.3	Eisenanalyse met use case-diagrammen	307

7.4	Van use case naar activiteitendiagram (workflow)	311
7.5	Van activiteitdiagram naar sequentiediagram	312
7.6	Het klassendiagram (class diagram)	314
7.7	Object Oriented Programming	326
8	XML	345
8.1	Inleiding XML	345
8.2	XML en PHP	359

1.1 Inleiding HTML5

1.1.1 Evolutie van HTML

HTML of HyperText Markup Language is in 1991 ontwikkeld door sir Tim Berners-Lee om wetenschappelijke documenten van het CERN toegankelijk te maken. Al snel werden de mogelijkheden van HTML ondervonden en in de loop der jaren verschenen steeds betere browsers. Het World Wide Web Consortium (W3C) nam in 1996 de ontwikkeling van HTML over. De versie HTML 3.2 kwam in januari 1997 tot stand. In december 1997 volgde een eerste versie van de HTML 4-specificatie. Een eerste versie van HTML5 werd in januari 2008 gepubliceerd. Al in 2007 zei Steve Jobs dat HTML5 de software Flash overbodig zou maken. Hij herhaalde zijn woorden bij de introductie van de iPad in 2010.

De belangrijkste aanpassingen in HTML5 zijn: het uniform afhandelen van fouten, body- en header-tags gebruiken is niet nodig (maar wel aan te raden), de mogelijkheid om een document logischer op te bouwen met elementen als `<section>`, `<article>`, `<aside>` en `<nav>` en nieuwe invoertypes. HTML5 wordt ook ondersteund door alle nieuwe browsers, waaronder FireFox, Google Chrome en Safari.

1.1.2 HTML-editors

Om te kunnen beginnen met het coderen van webpagina's hebben we alleen een webbrowser en een HTML-editor nodig. Er zijn meerdere HTML-editors gratis te downloaden op internet. Een simpele editor is TotalEdit. Als je nog geen HTML-editor geïnstalleerd hebt, probeer dan TotalEdit te downloaden en installeren.

De URL van de site is www.totaledit.com.

1.1.3 Lay-out en design van een webpagina

Uit onderzoek blijkt dat de meeste webdesigners in de loop der jaren zijn gaan ontwerpen volgens dezelfde patronen.

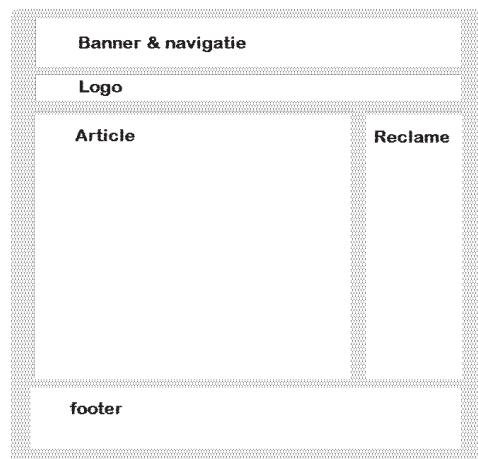
Bij de meeste moderne websites komen we het volgende ontwerp tegen:

- een webpagina met een banner en navigatie bovenaan;
- een of meer artikelen die de content vormen;
- een kolom aan de rechterkant voor reclame;
- een voettekst met bedrijfsinformatie onderaan.

Een voorspelbare website is makkelijker om te navigeren en de informatie is eenvoudiger te vinden.

Figuur 1.1

Lay-out van een webpagina



Figuur 1.2

Eindresultaat



In de volgende hoofdstukken gaan we een website maken voor de Marathon van Amsterdam. Dit project pakken we op de volgende manier aan. Ten eerste splitsen we het project in kleinere delen en ten tweede wisselen we bij ieder deel tussen markup (tags in de vorm van code) en design (ontwerp). Het eindresultaat moet er ongeveer als figuur 1.2 uitzien.

1.2 Meta-informatie

Meta-informatie is informatie over informatie. Webpagina's mogen informatie over de paginacontent bevatten. Deze informatie is belangrijk voor de browser en voor zoekmachines.

1.2.1 Tags of markups

Webpagina's bestaan uit platte tekst met markups (*markeringstekens*), ook wel aangeduid als *tags*. Hierin wordt de lay-out aangegeven, ofwel hoe de tekst gesstructureerd moet worden. De *tags* worden door de webbrowser geïnterpreteerd en de gebruiker krijgt de opgemaakte tekst in de vorm van een webpagina te zien. Een commando tussen haakjes, bijvoorbeeld `<html>` of `<form>`, wordt een *tag of markup* genoemd. De syntaxis van tags is als volgt:

```
<open tag> content van de tag </sluiten tag>
```

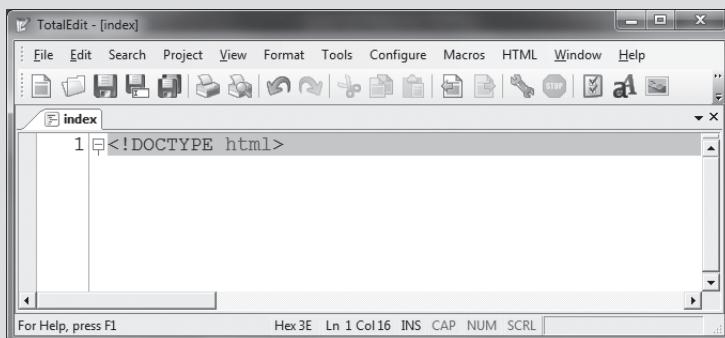
Als je een *begintag* hebt geopend, bijvoorbeeld `<html>` eindig je *altijd* met een *eindtag*, bijvoorbeeld `</html>`. Je ziet dat de *eindtag* altijd een / bevat.

1.2.2 Doctype

De eerste tag boven elk webdocument is de tag DOCTYPE. Deze tag is belangrijk voor de browser. Zo weet de browser wat voor type document dit is. Voor het volgende voorbeeld openen we een HTML-editor en beginnen met het typen van de eerste tag.

Opgave 1 Typ de tekst in de figuur over en sla deze op als **index.html**

Figuur 1.3
Opgave 1



Het resultaat van Opgave 1 is een document met de naam **index.html** en met de DOCTYPE html.

1.2.3 Het element <html>

Met het openen en sluiten van de <html>-tag maken we een webpagina. Alles wat tussen de <html>-tags in komt noemen we de content van de webpagina.

- Opgave 2** 2 Open **index.html** en voeg de volgende begin-html-tag en eind-html-tag eraan toe.

Figuur 1.4
Opgave 2

```

1  <!DOCTYPE html>
2  <html lang="nl">
3
4  </html>

```

Bij regel 2 hebben we het <html>-element als volgt gecodeerd:

```
<html lang= "nl">
```

We coderen de namen van alle tags met kleine letters.

Het attribuut `lang=` staat voor *language* of taal. We coderen de waarden van de attributen tussen aanhalingstekens:

```
lang = "nl"
```

1.2.4 Het element <head>

Het eerste wat we doen is de webpagina splitsen in twee onderdelen: een ‘kop’ en een ‘lichaam’. De kop of <head>-element gebruiken we voor de meta-informatie van het document. Het lichaam of <body>-element bevat de tags die de content van de webpagina bepalen. De content is dus de webpagina die zichtbaar wordt in de browser.

- Opgave 3** 3 Open **index.html** en voeg de volgende <head>-en <body>-tags toe.

Figuur 1.5
Opgave 3

```

1  <!DOCTYPE html>
2  <html lang="nl">
3      <head>
4
5      </head>
6      <body>
7      </body>
8  </html>

```

1.2.5 Het element <meta>

Binnen een meta-tag plaatsen we specifieke meta-informatie. Meta-informatie is belangrijk voor webbrowsers en zoekmachines zoals

Google en Yahoo. Zoekmachines gebruiken meta-informatie in hun algoritmes voor zoekopdrachten op internet. We kunnen meta-tags ook gebruiken voor het automatisch uploaden van een webpagina.

Iedere meta-tag krijgt twee attributen:

- name
- content

In het volgende voorbeeld coderen we een meta-tag om aan te geven wat de character encoding (tekencodering) van het document is.

Opgave 4 Open **index.html** en voeg de volgende meta-tags eraan toe.

```
<head>
    <meta charset="iso-8859-1">
    <meta name="description" content="marathon amsterdam">
    <meta name="keywords" content="marathon, resultaten,
        inschrijven">
    <meta name="author" content="jouw naam hier plaatsen">
</head>
```

Bij het vorige voorbeeld geeft de eerste meta-tag de charset (tekenset) aan. Voor webpagina's in Europese talen, bijvoorbeeld Nederlands, gebruiken we `charset= "iso-8859-1"` Daarnaast hebben we ook meta-tags met een beschrijving van de inhoud van de pagina en de naam van de auteur van de pagina.

Bij elementen zonder content zoals het `<meta>` gebruiken we geen eind-tag `< /meta>`

1.2.6 Het element `<title>`

Iedere webpagina krijgt een titel zodat tijdens het surfen op de website de titel van de pagina te zien is in de titelbalk van de browser. In het volgende voorbeeld gaan we de titel **Marathon van Amsterdam** aan onze webpagina toevoegen.

Opgave 5 Open **index.html** en voeg de volgende titel eraan toe:

```
<head>
    <meta charset="iso-8859-1">
    <meta name="description" content="marathon amsterdam">
    <meta name="keywords" content="marathon, resultaten,
        inschrijven">
    <meta name="author" content="jouw naam hier plaatsen">

    <title>Marathon van Amsterdam</title>

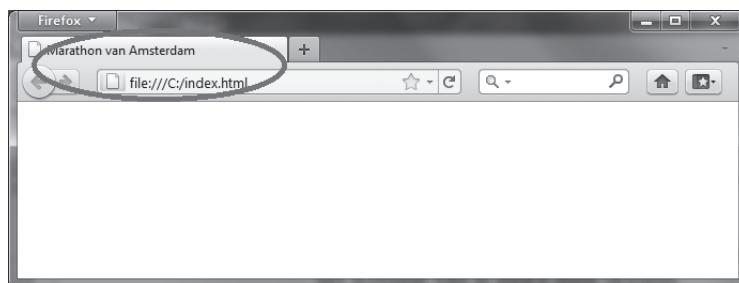
</head>
```

Met deze code heb je aangegeven dat er een titel `<title>` in de titelbalk van de browser moet komen. Dit kun je overigens testen door **index.html** te openen met de browser.

Open **index.html** met de browser door dubbel te klikken op het pictogram van het bestand **index.html**.

Figuur 1.6

Het resultaat van
Opgave 5



1.2.7 De <!-- commentaar -->-tag

De <!-- -->-tag doet feitelijk niks. Hij wordt alleen gebruikt om uitleg of commentaar te geven bij wat we aan het doen zijn. Bijvoorbeeld:

```
<!-- deze pagina is de homepagina -->
```

1.3 Pagina-markup

HTML5 maakt gebruik van de volgende tags om de content van een webpagina te structureren.

- <body>
- <div>
- <section>
- <header>
- <nav>
- <aside>
- <article>
- <footer>

1.3.1 Het element <body>

Content die je tussen de <body>-tag typt, zal zichtbaar worden in de browser. Hier bepalen we de markup van de pagina.

1.3.2 Het element <div>

Het <div>-element heeft geen semantische betekenis. We gebruiken <div> voor generieke content containers.

1.3.3 Het element <section>

Dit element gebruiken we voor het groeperen van verwante content. Content binnen een <section>-element heeft een bepaalde relatie met het thema van het <section>-element. Bijvoorbeeld, een <section> voor sport, nieuws of bedrijfsinformatie.

Opgave 6 Open **index.html** en voeg de volgende **<section>** eraan toe:

```
<body>
<section id="content">
</section>
</body>
```

1.3.4 Wat is een id?

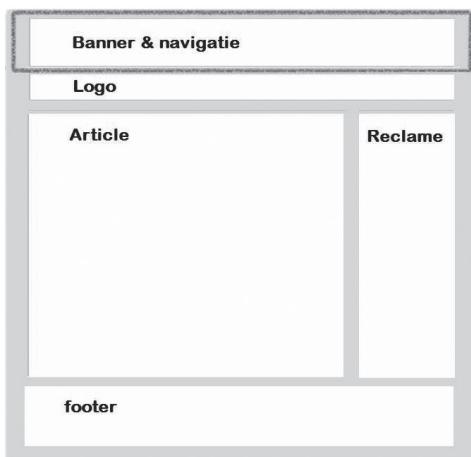
Het attribuut **id** identificeert een element met een naam. In dit geval is de naam van het **<section>**-element “**content**”.

1.3.5 De elementen **<header>** en **<hgroup>**

De eerste content in een pagina is het **<header>**-element. Dit element is bedoeld voor de banner, de inleidende informatie over de pagina en de navigatie van de site.

Figuur 1.7

De banner en navigatie

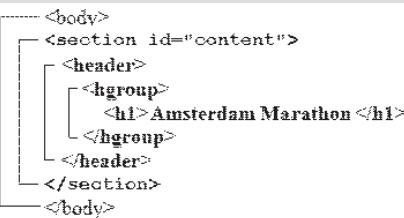


Opgave 7 Open **index.html** en voeg de elementen **<header>** en **<hgroup>** eraan toe:

```
<body>
<section id="content">
    <header>
        <hgroup>
            <h1>Amsterdam Marathom</h1>
        </hgroup>
    </header>
</section>
</body>
```

Je ziet dat een **<section>**-element een blok is en dat we binnen dit blok meerdere elementen mogen plaatsen. Als we een element binnen een element plaatsen gebruiken we de TAB-toets voor rechts inspringen. Op deze manier houden we onze codes leesbaarder. We kunnen duidelijk zien dat het **<header>**-element hoort bij het **<section id= "content">**-element.

Figuur 1.8
Inspringen met de TAB-toets.



1.3.6 Hiërarchie van elementen

De hiërarchie van de elementen wordt gevormd door parent- en child-elementen. Het `<html>`-element is het parent-element van alle andere elementen. Het `<body>`-element is een child-element van het `<html>`-element maar ook een parent-element van het `<section>`-element.

1.3.7 De elementen `<h1>` t/m `<h6>`

De elementen `<h1>` t/m `<h6>` gebruiken we voor kopteksten met een ge-definieerde lettergrootte van groot naar klein.

```

<h1> Amsterdam Marathon</h1>
<h2> Amsterdam Marathon</h2>
<h3> Amsterdam Marathon</h3>
<h4> Amsterdam Marathon</h4>
<h5> Amsterdam Marathon</h5>
<h6> Amsterdam Marathon</h6>

```

1.4 Navigatie markup

De navigatie van een website moet ervoor zorgen dat de gebruiker op een logische en eenvoudige manier alle informatie binnen de website kan vinden.

1.4.1 Het element `<nav>`

Binnen het `<header>`-element plaatsen we ook het `<nav>`-element. Hier gaan we elementen voor de navigatie van de site plaatsen.

Opgave 8 Open `index.html` en voeg het volgende `<nav>`-element eraan toe:

```

<header>
  <hgroup>
    <h1>Amsterdam Marathon</h1>
  </hgroup>
  <nav>
  </nav>
</header>

```

Voordat we beginnen met de navigatie van de site gaan we kijken naar twee soorten HTML-lijsten.

1.4.2 genummerde lijsten

Soms wil je een bepaald stuk tekst als lijst weergeven. Er bestaan twee soorten lijsten: genummerde lijsten en opsommingslijsten. We kijken eerst naar genummerde lijsten.

```
<ol>
    <li>Webdesign</li>
    <li>Webhosting</li>
    <li>Domeinregistratie</li>
</ol>
```

Bovenstaande tags genereren een lijst of groep `` met drie ``-lijst-items. Het resultaat van bovenstaande codes is de volgende genummerde lijst:

1. Webdesign
2. Webhosting
3. Domeinregistratie

1.4.3 opsommingslijsten

Een opsommingslijst is een lijst met bullets. Bijvoorbeeld, de volgende codes:

```
<ul>
    <li>HTML</li>
    <li>JavaScript</li>
    <li>PHP</li>
</ul>
```

genereren de volgende opsommingslijst:

- HTML
- JavaScript
- PHP

1.4.4 Het element <a>

De `<a>`-tag staat voor *anker* en wordt gebruikt om verschillende documenten met elkaar te verbinden. Verwijzen naar andere pagina's doe je op de volgende manier:

```
<a href="inschrijven.html">Inschrijven</a>
```

Het `<a>`-element heeft het attribuut `href` dat naar een webpagina verwijst. De content van het element is de tekst tussen de tags `<a>` en ``. In dit geval moet je op het woord **Inschrijven** klikken om naar de webpagina **inschrijven.html** te linken.

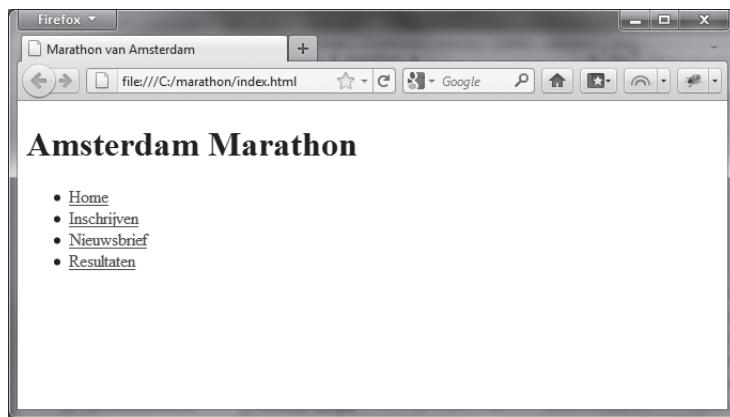
Opgave 9 Open **index.html** en voeg het volgende **<nav>**-element eraan toe:

```
<header>
    <hgroup>
        <h1>Amsterdam Marathon </h1>
    </hgroup>
    <nav>
        <ul>
            <li><a href="#top">Home</a></li>
            <li><a href="inschrijven.html">Inschrijven
                </a></li>
            <li><a href="nieuwsbrief.html">Nieuwsbrief
                </a></li>
            <li><a href="resultaten.html">Resultaten
                </a></li>
        </ul>
    </nav>
</header>
```

Open **index.html** met je browser en kijk naar je resultaten.

Figuur 1.9

Het resultaat van
Opgave 9



1.4.5 Code camp – Lab 1

In overleg met je docent kies je een wereldstad, bijvoorbeeld New York, Londen of Parijs. Als je eenmaal een stad gekozen hebt, begin je met het maken van een website voor de marathon van die stad. Het resultaat van Lab 1 moet er ongeveer zoals de vorige figuur uitzien.

1.5 Inleiding CSS3

CSS (Cascading Style Sheets) is ontstaan uit de onvrede onder web-designers over het gebruik van HTML-tags voor het opmaken van webpagina's. De CSS Working Group van de W3C heeft versie 1 van CSS al eind 1996 ontwikkeld. Het gaf designers voor het eerst de mogelijkheid om de opmaak van lettertypen en de positie van de elementen van een webpagina te bepalen.

In mei 1998 maakte CSS2 een eind aan de verschillende interpretaties tussen de browsers van Netscape en Internet Explorer.

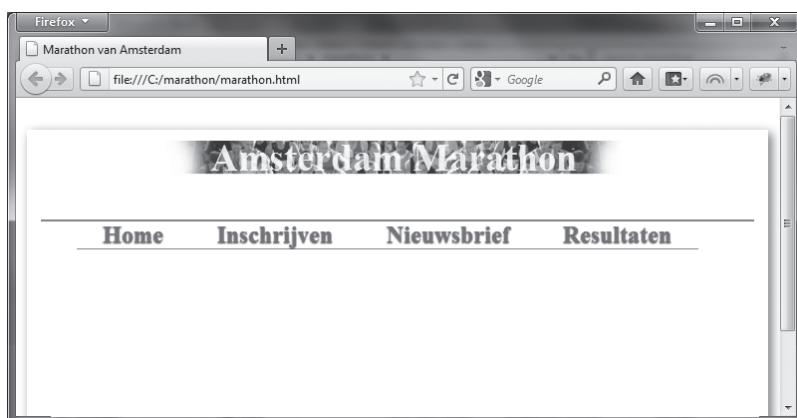
CSS3 biedt geavanceerdere mogelijkheden en zal het aantal afbeeldingen en JavaScript-elementen terugdringen en de totale omvang van een pagina zal kleiner worden, wat ten goede komt aan de ranking bij zoekmachines. Belangrijk zijn daarnaast de grafische mogelijkheden, zoals drop shadows, ronde hoeken, meerdere achtergronden, animaties en transparantie.

1.6 Navigatiedesign

Nu de structuur van de navigatie voltooid is, gaan we verder met CSS3 voor het design van deze elementen. We willen bijvoorbeeld de navigatie in een balk presenteren. Aan het einde van deze paragraaf moet je het volgende resultaat kunnen zien:

Figuur 1.10

Resultaat van
deze paragraaf



1.6.1 Het element <link>

Iedere webpagina kan gelinkt worden met externe bestanden, zoals JavaScript-bestanden of CSS-bestanden. Met JavaScript maken we webapplicaties en interactieve webpagina's. JavaScript wordt behandeld in het volgende hoofdstuk van dit boek. In het volgende voorbeeld maken we een link vanuit onze **index.html**-webpagina naar een extern CSS-bestand.

Opgave 10 Open **index.html** en voeg de volgende **<link>** binnen het **<head>**-element toe:

```
<head>
    <meta charset="iso-8859-1">
    <meta name="description" content="marathon
    amsterdam">
    <meta name="author" content="jouw naam hier
    plaatsen">
    <title>Amsterdam Marathon</title>
    <link rel="stylesheet" href="styles.css">
</head>
```

Binnen het `<head>`-element hebben we een `<link>`-element geplaatst. Het `<link>`-element heeft twee attributen: het attribuut `rel` geeft aan dat dit een link is naar een stylesheet terwijl het attribuut `href` verwijst naar het stylesheet-bestand. In deze paragraaf gaan we het bestand **styles.css** maken.

1.6.2 CSS-syntaxis

De CSS-syntaxis bestaat uit drie onderdelen:

Syntaxis: `selector {attribuut: waarde}`

- **selector:** is het HTML-element of de tag waarop je attributen wilt toepassen.
- **attribuut:** is het attribuut of de eigenschap die je wilt toepassen.
- **waarde** is de toegepaste waarde.

Bijvoorbeeld:

```
body {color: black}
```

Het element `body` krijgt het attribuut `color` met de waarde `black`.

1.6.3 Definiëren van id's

Het eerste wat we gaan doen in het CSS-bestand is de afmeting van het `<section id='content'>`-element bepalen. Om stijlen voor een **id** te definiëren beginnen we met het teken `#` gevolgd door de naam van de **id**. Daarna plaatsen we de attributen binnen accolades. Bijvoorbeeld:

```
#content{width: 800px;}
```

Hier bepalen we dat het `<section id= "content">`-element 800 pixels breed is.

1.6.4 De attributen width en height

Met het attribuut `width` en het attribuut `height` bepalen we de breedte en de hoogte van een element in pixels. Een pixel is een punt in het beeldscherm. Bijvoorbeeld, als we een deel van de bannerafbeelding vergroten zien we dat deze 300 pixels breed en 150 pixels hoog is. Deze afmeting geven we als volgt aan:

```
width: 300px;  
height: 150px;
```

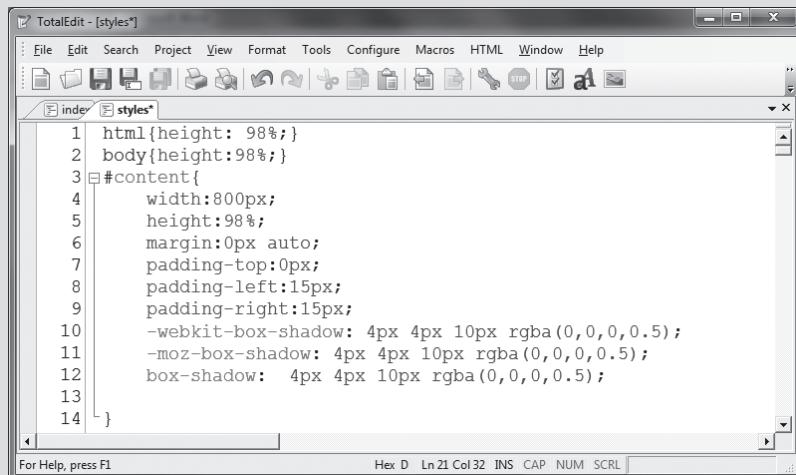
Figuur 1.11
Vergrote
afbeelding
in pixels



- Opgave 11** Open een nieuw bestand en sla het op als **styles.css**. Voeg de volgende stijlen eraan toe:

Figuur 1.12

Voorbeeld van
styles.css



```

TotalEdit - [styles.css]
File Edit Search Project View Format Tools Configure Macros HTML Window Help
index styles.css
1 html{height: 98%; }
2 body{height:98%; }
3 #content{
4     width:800px;
5     height:98%;
6     margin:0px auto;
7     padding-top:0px;
8     padding-left:15px;
9     padding-right:15px;
10    -webkit-box-shadow: 4px 4px 10px rgba(0,0,0,0.5);
11    -moz-box-shadow: 4px 4px 10px rgba(0,0,0,0.5);
12    box-shadow: 4px 4px 10px rgba(0,0,0,0.5);
13
14
For Help, press F1 Hex D Ln 21 Col 32 INS CAP NUM SCRL

```

Onze stijldefinitie geldt voor alle elementen met de id='content'. Deze elementen krijgen de volgende afmeting:

width: 800px	een breedte van 800 pixels
height: 98%	de hoogte is 98% van het browservenster
margin: 0px auto	een bovenste marge van 0 pixels
padding-top:0px	een ruimte van 0 pixels tussen de bovenzijde en de tekst
padding-left:15px	een ruimte van 15 pixels tussen de linkerzijde en de tekst
padding-right:15px	een ruimte van 15 pixels tussen de rechterzijde en de tekst

1.6.5 De attributen margin en padding

Het attribuut **margin** bepaalt de marges tussen de randen van elementen. Het attribuut **padding** is de afstand tussen de rand van het element en de inhoud binnen het element.

Figuur 1.13

Resultaat van
Opgave 11
in Firefox



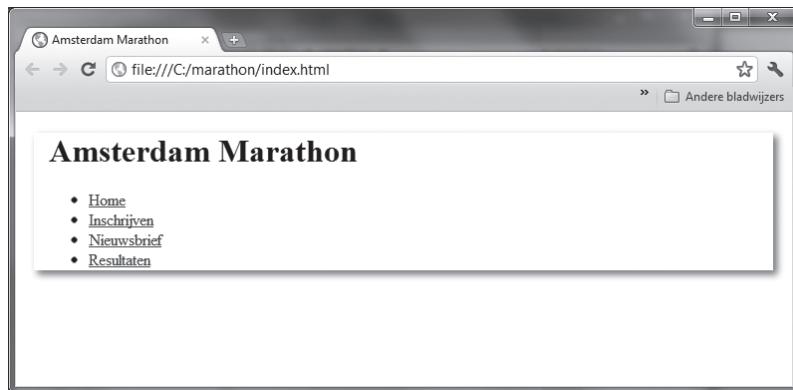
1.6.6 Cross-browser testen

Niet alle browsers hanteren dezelfde standaarden en niet alle browsers ondersteunen alle nieuwe HTML5-elementen of alle CSS3-stijlen. Het is daarom belangrijk dat we voortaan al onze scripts op meerdere browsers testen. In de volgende figuur zien we de presentatie van het `<div id='content'>`-element met de toegepaste afmeting uit het **styles.css** bestand.

Open **index.html** met de webbrowser Chrome en kijk naar het resultaat.

Figuur 1.14

Resultaat in
Chrome



1.6.7 Het attribuut box-shadow

Je ziet dat het `<div id='content'>`-element een attribuut `box-shadow` heeft gekregen. Box-shadow en text-shadow zijn nieuwe CSS3-attributen die ondersteund worden door Safari 3+, Chrome 3+, Firefox 3.5+, Opera 10.5+ en IE9. Het probleem is dat iedere browser eigen standaarden hantereert. In de volgende tabel zien we de browsers die HTML5 en CSS3 ondersteunen en het prefix voor iedere browser.

Tabel 1.1 Browsers en vendor-prefix

	Safari	<code>-webkit-</code>
	Mozilla	<code>-moz-</code>
	Opera	<code>-o-</code>
	Microsoft	<code>-ms-</code>
	Chrome	<code>-webkit-</code>

Voor ons betekent dit dat we deze attributen voor alle browsers moeten coderen. Bijvoorbeeld, het attribuut `box-shadow` moeten we als volgt coderen:

```
-webkit-box-shadow: 4px 4px 10px rgba(0,0,0,0.5);
-moz-box-shadow: 4px 4px 10px rgba(0,0,0,0.5);
box-shadow: 4px 4px 10px rgba(0,0,0,0.5);
```

In dit geval leest Safari alleen de eerste regel met het prefix

-webkit- en slaat de andere regels over. Alle browsers slaan de voor hen onbekende codes over.

```
-webkit-box-shadow: 4px 4px 10px rgba(0,0,0,0.5);
```

Mozilla-browsers lezen alleen de volgende regel met de prefix

```
-moz.
```

```
-moz-box-shadow: 4px 4px 10px rgba(0,0,0,0.5);
```

De laatste regel coderen we zonder prefix. Dit is de CSS3-standaard.

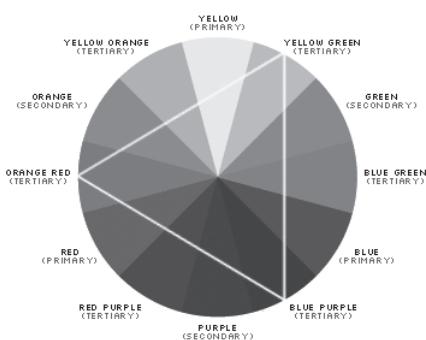
```
box-shadow: 4px 4px 10px rgba(0,0,0,0.5);
```

Het `<div id= 'content'>`-element krijgt dan een attribuut `box-shadow` van 4px aan de rechterzijde en 4px aan de onderzijde met een *blur* of vervaging van 10px.

1.6.8 Webkleuren

Webkleuren zijn gebaseerd op de basiskleuren rood, groen en blauw. Met een combinatie van deze drie kleuren creëren we nieuwe kleuren, zoals in onderstaande figuur.

Figuur 1.15
Kleurencirkel



1.6.9 Kleursyntax in CSS

CSS kent drie methoden om webkleuren te specificeren: decimaal, hexadecimaal en percentages.

De bekendste methode is hexadecimaal. Deze methode gebruikt drie hexadecimale nummers 0 t/m FF voor rood, groen en blauw.

Bijvoorbeeld:

```
rgb(00,00,FF)
```

Hier is de waarde voor rood 00 (of geen rood), de waarde voor groen is 00 (of geen groen) en de waarde voor blauw is FF (hoogste waarde). In andere woorden, alleen de kleur blauw heeft een waarde hoger dan nul.

Deze code correspondeert met de blauwe kleur in de eerste rij en de laatste kolom in onderstaande tabel.

Figuur 1.16

HTML-kleurcodes

Primary/Main/Basic Colors					
000000	000033	000066	000099	0000CC	0000FF
003300	003333	003366	003399	0033CC	0033FF
006600	006633	006666	006699	0066CC	0066FF
009900	009933	009966	009999	0099CC	0099FF
00CC00	00CC33	00CC66	00CC99	00CCCC	00CCFF
00FF00	00FF33	00FF66	00FF99	00FFCC	00FFFF
330000	330033	330066	330099	3300CC	3300FF
333300	333333	333366	333399	3333CC	3333FF
336600	336633	336666	336699	3366CC	3366FF
339900	339933	339966	339999	3399CC	3399FF
33CC00	33CC33	33CC66	33CC99	33CCCC	33CCFF
33FF00	33FF33	33FF66	33FF99	33FFCC	33FFFF

Bij de decimale methode worden de kleurwaarden weergegeven in decimalen van 0 t/m 255. Bijvoorbeeld:

`rgb(0, 255, 56)`

Bij de percentagemethode worden de kleurwaarden weergegeven in percentages van 0 t/m 100. Bijvoorbeeld:

`rgb(0%, 100%, 22%).`

1.6.10 Het attribuut `rgba`

Voor transparante kleuren gebruikt CSS3 het attribuut `rgba`. Met dit attribuut geef je na de kleurwaarden ook een transparantiewaarde op tussen 0.0 en 1.0 aan. Bijvoorbeeld:

`rgba(0, 255, 0, 0.3)`

specificeert de kleur groen met een transparantie van 30%

In ons geval krijgt het `<div id= 'content'>`-element een zwarte box-shadow met 50% transparantie:

`box-shadow: 4px 4px 10px rgba(0,0,0,0.5);`

1.6.11 Het attribuut `text-align`

Binnen het `<div id= 'content'>`-element hebben we een `<header>`-element met daarin een `<h1>`-element en een `<nav>`-element. In de volgende stap gaan we het `<header>`-element centreren. Hiervoor gebruiken we het attribuut `text-align`. De verschillende mogelijkheden zijn:

- links uitlijnen
- centreren
- rechts uitlijnen

Opgave 12 Open **styles.css** en voeg de volgende codes eraan toe:

```
header{  
    text-align:center;  
}  
  
hgroup{  
    background:url(images/banner.png);  
    background-position:center;  
    background-repeat:no-repeat;  
}
```

1.6.12 Het attribuut background

In het vorige voorbeeld krijgt het `<hgroup>`-element een attribuut `background`. Met dit attribuut hebben we een achtergrondafbeelding voor de banner toegevoegd.

```
background:url(images/banner.png);  
background-position:center;  
background-repeat:no-repeat;
```

De `background:url` is de locatie van de afbeelding. In dit voorbeeld is de afbeelding `banner.png` in de submap `images` geplaatst.

1.6.13 Background-position: en background-repeat:

```
background-position:center;  
background-repeat:no-repeat;
```

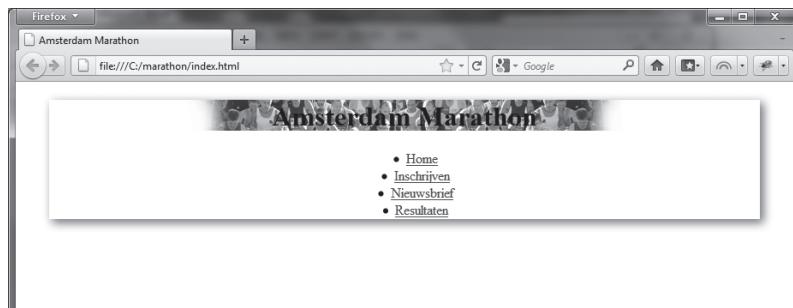
Background-position kan de volgende waarden hebben:

```
top | center | bottom  
top left | top center | top right  
bottom left | bottom center | bottom right
```

Met het attribuut `background-repeat` geef je aan of de afbeelding meerdere malen op de achtergrond moet worden getoond. In ons geval willen we geen herhaling.

Figuur 1.17

Resultaat van
Opgave 12



1.6.14 De attributen color en font-weight

Opgave 13 Open **styles.css** en voeg de volgende codes eraan toe:

```
hgroup h1 {
    color: rgba(255,255,255,0.8);
    font-family:serif;
    font-size:44px;
    font-weight: bold;
    text-align: center;
}
```

In het vorige voorbeeld krijgt de tekst **Amsterdam Marathon** in het **<h1>**-element de kleur wit met een transparantie van 80%, het lettertype serif, tekengrootte 44px, de tekst is vet en gecentreerd.

1.6.15 Het attribuut font-size

Met het attribuut **font-size** kunnen we in pixels, punten, percentages of veelvoud coderen.

Tabel 1.2 tekengrootte tekst

metingen	keyword	voorbeeld
Pixels	px	font-size : 10px;
Points	pt	font-size : 20pt ;
Percentage	%	font-size : 30%;
Veelvoud	em	font-size : 1.5em;

De meest exacte aanduiding voor font-size (tekengrootte) is in pixels. Voor gebruikersvriendelijke afdrukken gebruiken we points. Percentages en veelvoud zijn font-sizes die in een vaste verhouding staan tot de font-size van het parent-element.

Bijvoorbeeld:

```
body {font-size : 10px;}
body h1 {font-size : 2.0em;}
body h2 {font-size : 50%;}
```

Het **<body>**-element is het parent-element met een font-size van 10 pixels.

Het **<h1>**-element is het child-element met twee maal de font-size van het parent-element (20 pixels).

Het **<h2>**-element is ook een child-element met 50 procent van de font-size van het **<body>**-element (5 pixels).

1.6.16 Het attribuut border

Borders zijn de randen van de elementen. Met het attribuut `border` bepalen we de stijl, kleur en dikte van de borders.

Opgave 14 Open `styles.css` en voeg de volgende codes eraan toe:

```
nav ul {  
    border-top: double 3px #8e8e8e;  
    text-align: center;  
}  
nav ul li{  
    float:left;  
    padding:0px 30px;  
    list-style-type:none;  
    border-bottom: solid 1px #8e8e8e;  
}
```

In dit voorbeeld krijgt de bovenste border van de ``-opsomminglijst binnen het `<nav>`-element de stijl double, de dikte 3px en de hexadecimale kleur `#8e8e8e`

1.6.17 Het attribuut float

Float (drijven), wordt gebruikt voor het links of rechts uitlijnen van elementen. In het vorige voorbeeld krijgen alle ``-lijstitems het attribuut `float`. Met dit attribuut verplaatsen we alle lijstitems binnen het `<nav>`-element naar links. Het is alsof de elementen een magneet maar links krijgen. Eerst het item Home gevolgd door de items Inschrijven Newsbrief en Resultaten.

Met de volgende regel:

```
list-style-type:none;
```

halen we de bulletpoints of opsommingtekens weg.

Figuur 1.18

Resultaat van
Opgave 14



1.6.18 Het attribuut border-radius

Het attribuut `border-radius` zorgt voor ronde hoeken van elementen. In ons geval willen we dat alle hyperlinks van de navigatie ronde borders krijgen.

Opgave 15 Open **styles.css** en voeg de volgende border-radius-codes eraan toe:

```
nav ul li a
{
    font-size:20pt;
    font-weight: bold;
    color :rgb(86, 122, 199);
    text-decoration: none;
    text-shadow: 0 1px 1px rgba(0,0,0,0.70);
    -webkit-border-radius:6px;
    -moz-border-radius:6px;
    border-radius:6px;
}
```

1.6.19 Het attribuut transition

Als de bezoeker met de muis over de navigatiekoppelingen beweegt, willen we een animatie-effect creëren. Daarvoor gebruiken we het attribuut **transition**.

Opgave 16 Open **styles.css** en voeg de volgende transition-codes eraan toe:

```
nav ul li a
{
    font-size:20pt;
    font-weight: bold;
    color :rgb(86, 122, 199);
    text-decoration: none;
    text-shadow: 0 1px 1px rgba(0,0,0,0.70);
    -webkit-border-radius:6px;
    -moz-border-radius:6px;
    border-radius:6px;
    -webkit-transition:all 1.0s ease-in-out;
    -moz-transition:all 1.0s ease-in-out;
    -o-transition:all 1.0s ease-in-out;
    transition:all 1.0s ease-in-out;
}
```

Bovenstaande stijlen gelden alleen voor de `<a>`-hyperlinks binnen een ``-lijstitem, binnen een ``-lijst binnen het `<nav>`-element. Hier willen we een animatie-effect van **1.0s** seconde. Deze animatie wordt door de muis geactiveerd met een **ease-in-out** effect.

1.6.20 De pseudoklassen hover en focus

Een `<a>`-hyperlink heeft vier pseudoklassen:

- `hover`
- `focus`
- `visited`
- `active`

Pseudoklassen van het `<a>`-element definiëren we als volgt:

```
a:hover
a:focus
```

Opgave 17 Open **styles.css** en voeg de volgende codes eraan toe:

```
nav ul li a:hover,
nav ul li a:focus
{
    color: #fff;
    background: rgb(86,122,199);
}
```

Op het moment dat de bezoeker met de muis over de hyperlinks van de navigatie beweegt verandert de tekstkleur van blauw naar wit en de achtergrondkleur van wit naar blauw. De animatie duurt een seconde.

Figuur 1.19

Resultaat van 17



1.6.21 Code camp – Lab 2

In overleg met je docent heb je een wereldstad gekozen zoals New York, Londen, Parijs, etc. In dit lab ga je de designstyles die je in deze paragraaf geoefend hebt toepassen op je webpagina uit Lab 1. Ontwerp je eigen navigatie. Gebruik andere kleuren.

1.7 Logo markup

We beginnen deze paragraaf met een stadslogo. In dit geval maken we een logo voor de stad Amsterdam. Met dit logo kunnen we in de volgende paragraaf illustreren hoe lagen (*layers*) in CSS3 in zijn werk gaan.

Opgave 18 Open **index.html** en voeg na het **<header>**-element de volgende elementen eraan toe:

```
<div id="intro">
    <div id="rodelogo" class="logo">
        <span id="iam">I am</span>sterdam</div>
    <div id="blauwelogo" class="logo">
        <span id="iam">I am</span>sterdam</div>
    <div id="groenelelogo" class="logo">
        <span id="iam">I am</span>sterdam</div>
    </div>
```

In dit voorbeeld hebben we de klasse `class="logo"` drie keer gecodeerd.

1.7.1 Class

Als je een bepaalde stijl meerdere keren wil toepassen is het raadzaam om een klasse te maken. In het vorige voorbeeld hebben we drie nieuwe div's gecodeerd met `class="logo"`

```
<div id="rodelogo" class="logo">
<div id="blauwelogo" class=" logo ">
<div id="groenelogo" class=" logo ">
```

1.7.2 Het element

Het ``-element gebruiken we op tektniveau. Zo kunnen we een deel van een tekst als een element definiëren. Bijvoorbeeld:

```
<span id="iam">I am</span>sterdam
```

In dit voorbeeld heeft het eerste deel “I am” van de tekst

“I am sterdam”

de id= “iam” gekregen.

Figuur 1.20

Het element



1.8 Logo design

In deze paragraaf gaan we een logo ontwerpen met de tekst:

“I amsterdam”

We beginnen met de afmeting en met het positioneren van het element `<div id= "intro">`.

1.8.1 Het attribuut position

Alle elementen mogen het attribuut `position` hebben. Met dit attribuut kunnen we elementen in het beeldscherm exact positioneren.

Als de positie in `relative`-termen wordt gegeven dan betekent dat een positie ten opzichte van de borders van het `parent`-element.

Als de positie in `absolute` termen wordt gegeven dan betekent dat een exacte positie ten opzichte van de borders van het `<body>`-element.

Opgave 19 Open `styles.css` en voeg de volgende designattributen eraan toe:

```
#intro{
    height:80px;
    margin:0px 0px 0px 0px;
    position:relative;
    top:20px;
    left:20px;
}
```

```
#iam{  
    font-family:segoe ui;  
    font-size:1.0em;  
    color:#fff;  
}
```

In dit voorbeeld heeft de tekst “I am” in het ``-element een witte tekstkleur gekregen.

1.8.2 Definiëren van klassen

Om stijlen voor een klasse te definiëren beginnen we met het punt teken (.) gevuld door de naam van de klasse. Bijvoorbeeld:

```
.logo{ font-family:segoe ui;}
```

Opgave 20 Open `styles.css` en voeg de volgende designattributen eraan toe:

```
.logo{  
    font-family:segoe ui;  
    font-size:2.0em;  
    color:#000;  
    padding-left:30px;  
    margin:0px 0px 0px 0px;  
    border-radius:40px;  
    -moz-border-radius:40px;  
    -webkit-border-radius:40px;  
    opacity:0.5;  
    -ms-filter:"progid:DXImageTransform.Microsoft.  
    Alpha(Opacity=50)";  
}
```

1.8.3 Het attribuut border-radius

Met het attribuut `border-radius` bepalen we de rondte van de borders van een element. In het vorige voorbeeld hebben alle logo's ronde hoeken gekregen:

```
border-radius:40px;
```

1.8.4 Het attribuut opacity

Met het attribuut `opacity` kunnen we elementen transparant weergeven. In het vorige voorbeeld hebben alle logo's een transparantie van 50% gekregen. De laatste regel is gecodeerd voor de webbrowser Internet Explorer.

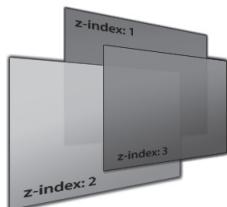
```
opacity:0.5;  
-ms-filter:"progid:DXImageTransform.Microsoft.  
Alpha(Opacity=50);
```

Met het attribuut `opacity` maken we elementen transparant. Met het attribuut `rgba` maken we kleuren transparant.

1.8.5 Het attribuut z-index

Met het attribuut `z-index` kunnen we layers (lagen) definiëren. Elementen met een hoger `z-index`-getal verschijnen op de voorgrond en elementen met een lager `z-index`-getal worden op de achtergrond vertoond. Dat betekent dat we elementen kunnen laten overlappen.

Figuur 1.21
Lagen met
`z-index`



Hieronder krijgen alle logo's een eigen `z-index`

Opgave 21 Open `styles.css` en voeg de volgende designattributen eraan toe:

```
#rodelogo{
    z-index: 1;
    width: 250px;
    height: 60px;
    background: rgb(246, 141, 145);
}
#blauwelogos{
    z-index: 2;
    position: relative;
    top: -70px;
    left: 60px;
    width: 250px;
    height: 80px;
    background: rgb(159, 163, 229);
}
#groenelogos{
    z-index: 3;
    position: relative;
    top: -100px;
    left: 120px;
    width: 200px;
    height: 60px;
    background: rgb(144, 216, 165);
}
```

Figuur 1.22
Opgave 21

The screenshot shows a website header with the text 'Amsterdam Marathon'. Below the header is a navigation menu with links: 'Home', 'Inschrijven', 'Nieuwsbrief', and 'Resultaten'. Underneath the menu are three circular buttons, each containing the text 'I Amsterdam'. The buttons are semi-transparent and overlap each other, demonstrating the effect of the `z-index` property. The button furthest to the right has the highest `z-index` value and is therefore positioned in front of the others.

1.8.6 Code camp – Lab 3

In dit lab ga je de markup en het design van een logo voor je webpagina maken. Ontwerp je eigen logo.

1.9 Article markup

Iedere pagina moet een hoofdartikel hebben. In deze paragraaf gaan we onze **index.html** verder uitbreiden. Na de navigatie en het logo willen we nu het hoofdartikel van de pagina structureren. Dit doen we met het element `<article>`.

1.9.1 Het element `<article>`

Als we naar onze begin lay-out kijken zien we dat de namen van de elementen voor de header, nav, footer en aside beschrijvingen zijn van de content van het element. Het `<article>`-element is een element dat *self-contained* is en deel uitmaakt van een `<section>`-element. Bijvoorbeeld, in een `<section id = "sport">`-element zouden we meerdere `<article>`-elementen over sport kunnen plaatsen. In dit geval maakt het `<article>`-element deel uit van het `<section id= "content">`-element. Een `<article>`-element begint met een `<header>`-element.

1.9.2 Het element `<p>`

Met het `<p>`-element openen we een nieuwe alinea en met `</p>` sluiten we de alinea. Alle tekst ertussenin is de inhoud van de alinea. Op alinea's kunnen we allerlei tekstattributen toepassen, zoals lettertype, tekengrootte, vet, cursief en links, center en rechts uitlijnen.

Bijvoorbeeld:

```
<p>De tekst van de alinea</p>
```

1.9.3 Inline- of tekstelementen

In alinea's mag je de volgende elementen gebruiken:

- strong
- em
- mark
- abbr

1.9.4 Het element ``

Het ``-element gebruiken we om een woord of tekst vet weer te geven.

Bijvoorbeeld:

```
<strong>I Amsterdam</strong>
```

1.9.5 Het element ``

Het ``-element gebruiken we om een woord of tekst cursief weer te geven.

Bijvoorbeeld:

```
<em>I Amsterdam</em>
```

1.9.6 Het element <mark>

Het <mark>-element gebruiken we om een woord of tekst geel markeren. Bijvoorbeeld:

```
<mark>I Amsterdam</mark>
```

1.9.7 Het element <abbr>

Het <abbr>-element gebruiken we om een verkorte tekst te ondertitelen, zodat je de oorspronkelijke tekst te zien krijgt als je met de muis over de afkorting beweegt.

Bijvoorbeeld:

```
<abbr title= "Verenigde Naties" >VN</abbr>
```

Opgave 22 Open **index.html** en voeg het <article>-element er aan toe:

```
<article class="rondehoeken">
<header>
    <div class="streep1"></div>
    <div class="streep2"></div>
    <div class="streep3"></div>
    <div class="streep4"></div>
    <div class="streep5"></div>
    <h1>Artikeltitel</h1>
</header>
<div id="artikel-container">
<p id="item">Aliquam erat volutpat. <em>Mauris</em> vel neque sit amet nunc gravida <strong>congue sed sit amet purus</strong>. <mark>Quisque</mark> lacus quam, egestas ac tincidunt a, lacinia vel velit.<br />
In <abbr title="kruiderij">condimentum</abbr> orci id nisl volutpat bibendum. <mark>Quisque</mark> commodo hendrerit lorem quis egestas. Vivamus rutrum nunc non neque consectetur quis placerat neque lobortis. Nam vestibulum, arcu sodales feugiat consectetur, nisl orci bibendum elit, eu euismod magna sapien ut nibh. Aliquam erat volutpat. Mauris vel neque sit amet nunc gravida congue sed sit amet purus.</p>
</div> <!--artikel-container -->
</article>
```

Figuur 1.23

Resultaat uit voorbeeld

The screenshot shows a web page with a header containing five decorative horizontal bars. The main content is an article with a paragraph of Latin text. Several words in the text are highlighted with a yellow background and a black border, demonstrating the use of the <mark> element. The highlighted words include 'Amsterdam' multiple times and other words like 'Mauris', 'Vivamus', etc. Below the paragraph is a heading 'Artikeltitel'. At the bottom of the page, there is a footer area with more Latin text.

Hier is het woord Mauris cursief weergegeven, het woord Quisque geel gemarkeerd, de tekst “congue sed sit amet purus” vet weergegeven en als je met de muis over het woord condimentum gaat dan krijg je de ondertiteling “kruiderij”.

1.9.8 Code camp – Lab 4

Open je eigen marathonproject en voeg een `<article>`-element eraan toe. Het resultaat moet er ongeveer zoals in de vorige figuur uitzien.

1.10 Article design

In deze paragraaf kijken we naar de volgende attributen:

- overflow
- gradient
- text-shadow
- Hue (H), Saturation (S) en Lightness (L)
- line-height

1.10.1 Het attribuut overflow

We willen een vaste afmeting voor het `<article>`-element. Dit betekent dat de content groter kan zijn dan de afmeting van het `<article>`-element. Wanneer dit gebeurt noemen we dat overflow. Met het attribuut `overflow` zorgen we ervoor dat een scrollbar, ook wel schuifbalk verschijnt:

```
overflow: scroll;
```

1.10.2 De function gradient()

Voor het `<article>`-element willen we een lichtblauwe achtergrondarcing. Hieronder definiëren we de afmetingen en de achtergrondarcing voor het `<article>`-element.

Opgave 23 Open `styles.css` en voeg de volgende stijlen er onderaan aan toe:

```
article{  
    float: left;  
    width: 540px;  
    height: 500px;  
    margin: 10px;  
    overflow: scroll;  
    padding: 0 0 0 0;  
    border: 1px solid #ddd;  
    background: -moz-linear-gradient(top, #94bae7 0%, #fff  
    100%);  
    background: -webkit-gradient(linear, left top, left  
    bottom, from(#94bae7), to(#fff));  
    background: -o-gradient(#94bae7 , #fff);  
}
```

Het `<article>`-element heeft de opgegeven afmetingen en een lichtblauwe achtergrondarcering gekregen. Het `<article>`-element maakt gebruik van de klasse ronde hoeken. Deze klasse moet ervoor zorgen dat het element ronde hoeken krijgt.

Opgave 24 Open **styles.css** en voeg de volgende stijlen onderaan toe:

```
.rondehoeken{
    border-radius:10px;
    -moz-border-radius:10px;
    -webkit-border-radius:10px;
}
```

1.10.3 Het attribuut `text-shadow`:

In het volgende voorbeeld definiëren we de stijlattributen voor het `<header>`-element binnen het `<article>`-element.

Opgave 25 Open **styles.css** en voeg de volgende designattributen eraan toe:

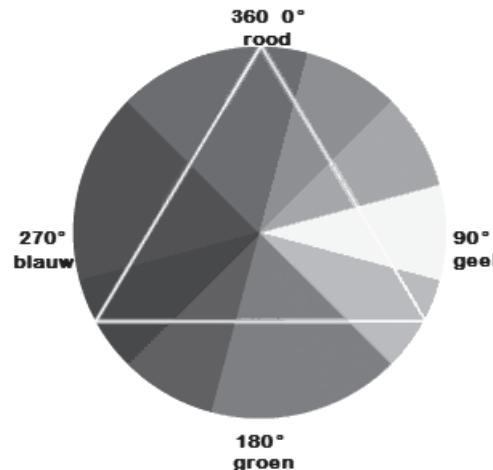
```
article header h1{
    z-index: 1;
    position:relative;
    top:-80px;
    left:30px;
    margin:10px 0px 0px 0px;
    font-family:Arial;
    font-size:1.5em;
    color:#fff;
    text-shadow:5px 5px 10px #000;
}
```

1.10.4 Het attribuut `hsla`

Met het attribuut `hsla` specificeren we tint (hue), verzadiging (saturation) helderheid (lightness) en transparantie (alpha)

Figuur 1.24

Het attribuut `hsla`



Hue is afgeleid uit de kleurcirkel: 0 t/m 360 graden.

Saturation is een percentage: vanaf grijstinten 0% tot volle kleur 100%

Lightness is een percentage: vanaf donker 0% tot lichtst 100%

Alpha is opacity (transparantie): vanaf onzichtbaar 0.0 tot zichtbaar 1.0

Bijvoorbeeld:

```
background:hsla(165, 35%, 50%, 1.0)
```

De kleurtint is groen

De verzadiging (grijstint) is 35%

De helderheid is 50%

De transparantie is 1.0

Opgave 26 Open **styles.css** en voeg de volgende designattributen eraan toe:

```
div.steep1 { background:hsla(165, 35%, 50%, 1.0);  
height:20px; }  
div.steep2 { background:hsla(165, 35%, 50%, 0.8);  
height:20px; }  
div.steep3 { background:hsla(165, 35%, 50%, 0.6);  
height:20px; }  
div.steep4 { background:hsla(165, 35%, 50%, 0.4);  
height:20px; }  
div.steep5 { background:hsla(165, 35%, 50%, 0.2);  
height:20px; }
```

1.10.5 Het attribuut line-height

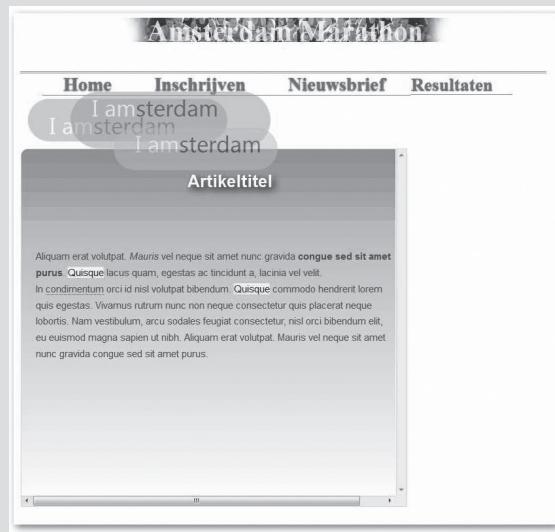
Met het attribuut `line-height` specificeren we de regelafstand van een tekst om ervoor te zorgen dat teksten leesbaar zijn.

Opgave 27 Open **styles.css** en voeg de volgende designattributen eraan toe:

```
#item {  
margin:0px 0px 20px 0px;  
font-family:Arial;  
font-size:0.88em;  
color:#4c4c4c;  
line-height:1.6em;  
}
```

Het resultaat moet er zoals de volgende figuur uitzien.

Figuur 1.25
Het attribuut line-height



1.10.6 Code camp – Lab 5

Open je eigen marathonproject en voeg de designstyles voor het element `<article>` eraan toe. Ontwerp je eigen achtergrond kleurschema.

1.11 Aside markup

Aside (opzij) gebruiken we voor de rechterkolom waar we additionele en interessante elementen voor onze website plaatsen. In deze paragraaf maken we een `<aside>`-element met een video en twee reclame-elementen.

1.11.1 Het element `<aside>`

Met het `<aside>`-element definiëren we een rechterkolom voor reclame of andere additionele informatie die zijdelings te maken heeft met de overige inhoud van de website. In het volgende voorbeeld plaatsen we een videofilmpje over de Marathon van Amsterdam binnen het `<aside>`-element.

Opgave 28 Open **index.html** en voeg het volgende `<aside>`-element onderaan het `<article>`-element eraan toe:

```

<aside>
<div id="video-container" >
<video controls width="150" height="250" preload id="videos"
poster="images/videoposter.jpg">

<source src='videos/wildlife.webm' type='video/webm'/>
<source src='videos/wildlife.ogv' type='video/ogg'/>
<source src='videos/wildlife.mp4' type='video/mp4'/>

<!-- fallback to Flash: -->
<object type="application/x-shockwave-flash" width="150"
height="250"
data="player.swf" >
<param name="movie" value="player.swf" >

```

```
<param name="allowFullScreen" value="true" >
<param name="flashvars"
value="config={'clip':{'url':'videos/wildlife.
mp4','autoPlay':false}}">

</object>
</video>
</div> <!-- /#video-container -->
</aside>
```

Figuur 1.26

Het element
<aside>



1.11.2 Het element <video>

Met het element <video> zetten we de attributen voor het videoobject op. Het attribuut poster geeft het beginbeeld van het video-object aan.

```
<video controls width="150" height="250" preload id="videos"
poster="images/videoposter.jpg" >
```

1.11.3 Het element <source>

Met het element <source> geven we aan waar de video te vinden is en wat voor type video het is.

```
<source src='videos/filmpje.webm' type='video/webm'/>
<source src='videos/filmpje.ogv' type='video/ogg'/>
<source src='videos/filmpje.mp4' type='video/mp4'/>
```

1.11.4 Videoformaat

HTML5 maakt het mogelijk om video's af te spelen zonder browser-plugins. Het probleem is echter dat er op dit moment nog geen standaardfor-

maat is. Niet alle browsers ondersteunen elk formaat. De indeling MP4 wordt op dit moment het meest gebruikt. Voor ons betekent dit dat we onze video's moeten converteren naar alle formaten.

Tabel 1.3 Browsers en videoformats

	Safari	MP4, Flash
	Mozilla	OGG, WebM, Flash
	Opera	OGG, WebM
	Microsoft	Flash
	Chrome	MP4, OGG, WebM, Flash

1.11.5 Videoconversie

Op internet is gratis software voor videoconversies te downloaden. Zoek een filmpje voor je website en maak daarna met behulp van een conversiesoftware meerdere filmpjes in de verschillende formaten voor alle browsers. Zorg ervoor dat je je video in ieder geval naar de volgende drie formaten converteert.

```
<source src='videos/filmpje.webm' type='video/webm'/>
<source src='videos/filmpje.ogv' type='video/ogg'/>
<source src='videos/filmpje.mp4' type='video/mp4'/>
```

1.11.6 Het element

We willen twee advertenties in het `<aside>`-element plaatsen. De eerste is voor het recyclen van oude schoenen en de tweede voor energierepen. Voorlopig plaatsen we een hyperlink naar dezelfde **index.html**-pagina.

```
<a href="#">
```

Opgave 29 Open **index.html** en voeg het volgende `<aside>`-element onder aan het `<video>`-element toe:

```
<div id="advert1" class="rondehoeken" >
    <a href="#"><br /><br />Recyclen!<br /><br /></a>
</div>

<div id="advert2" class="rondehoeken">
    <a href="#" ></a>
</div>
```

Het resultaat moet er zoals de volgende figuur uitzien.

Figuur 1.27
Advertenties
toevoegen



1.11.7 Code camp – Lab 6

Open je eigen marathonproject en voeg een `<aside>`-element met een `<video>`-element en twee advertenties eraan toe. Maak je eigen reclame.

1.12 Aside design

De attributen voor het `<aside>`-element gebruiken we om het `<video>`-element en de advertenties te positioneren.

Opgave 30 Open `styles.css` en voeg de volgende designattributen eraan toe:

```
aside {
    float: right;
    width: 150px;
    margin: 10px;
    padding: 10px;
    border: 1px solid #ddd;
}

#video-container {
    margin-left: 12px;
    margin-bottom: 12px;
}
```

1.12.1 Het attribuut `text-transform`

Met het attribuut `text-transform` kunnen we een tekst converteren naar hoofdletters, kleine letters of kleinkapitalen.

Bijvoorbeeld:

```
h1 {text-transform:uppercase}  
h2 {text-transform:capitalize}  
p {text-transform:lowercase}
```

1.12.2 Het attribuut background-position

Een achtergrondafbeelding positioneren doen we met het attribuut background-position. De eerste is de horizontale waarde en de tweede is de verticale waarde. De mogelijke waarden zijn:

```
left top  
left center  
left bottom  
right top  
right center  
right bottom  
center top  
center center  
center bottom
```

We kunnen ook exacte of relatieve waarden aangeven. Zoals hieronder:

```
300px 100px  
of  
50% 30%
```

Bijvoorbeeld:

```
background-image:url(images/shoe.jpg);  
background-repeat:no-repeat;  
background-position: 50% 88%;
```

Opgave 31 Open **styles.css** en voeg de volgende designattributen eraan toe:

```
#advert1 {  
  
background-image:url(images/shoe.jpg);  
background-repeat:no-repeat;  
background-position: 50% 88%;  
text-align: center;  
text-transform: uppercase;  
margin: 10px 5px;  
}  
  
#advert2{  
padding-top:30px;  
height: 100px;  
background: -moz-linear-gradient(top, #487a77 0%, #fff  
100%);  
background: -webkit-gradient(linear, left top, left bottom,  
from(#487a77), to(#fff));  
background: -o-gradient(#487a77 , #fff);  
margin: 10px 5px;  
}
```

```
#snickers{  
    opacity:0.5;  
    -ms-filter:"progid:DXImageTransform.Microsoft.  
    Alpha(Opacity=50);  
}
```

1.12.3 De functie rotate()

Met de functie `rotate()` kunnen we voor hyperlinks een rollover-effect voor de muis creëren.

Opgave 32 Open **styles.css** en voeg de volgende designattributen eraan toe:

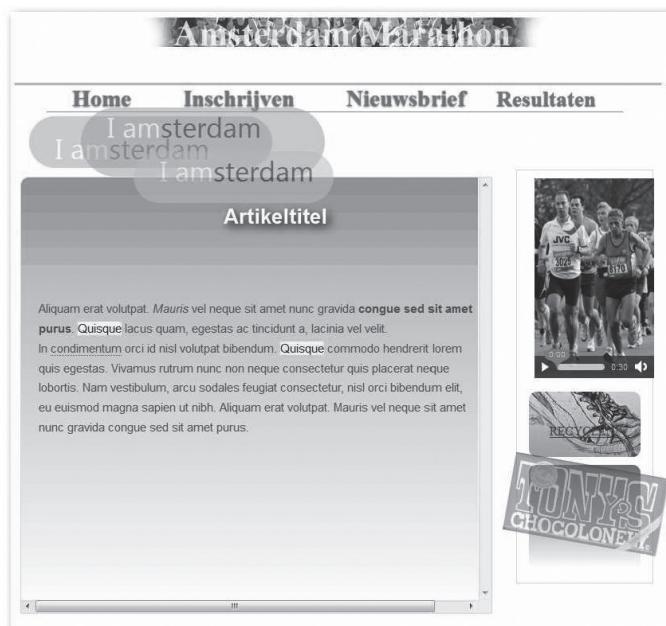
```
#advert2 a:hover{  
    color: #484848;  
    -webkit-transform:rotate(10deg) scale(1.5);  
    -moz-transform:rotate(10deg) scale(1.5);  
    -ms-transform:rotate(10deg) scale(1.5);  
    -o-transform:rotate(10deg) scale(1.5);  
    transform:rotate(10deg) scale(1.5);  
}
```

1.12.4 De functie scale()

De functie `scale(1.5)` betekent dat als je met de muis over de afbeelding beweegt, de afbeelding anderhalf keer vergroot wordt. De functie `scale(2.0)` zal de afbeelding twee keer zo groot maken. De functie `rotate(10deg)` zal de afbeelding een rotatie van 10 graden met de klok mee geven. Het resultaat moet er zoals de volgende figuur uitzien.

Figuur 1.28

De functie scale()



1.12.5 Code camp – Lab 7

In dit lab ga je de designstyles die je in deze paragraaf geoefend hebt toe-passen op je webpagina uit Lab 6. Ontwerp je eigen reclame.

1.13 Footer markup

Onder aan een pagina plaatsen we een `<footer>`-element.

1.13.1 Het element `<footer>`

In het `<footer>`-element kunnen we informatie over het bedrijf, de auteur en het copyright plaatsen.

- Opgave 33** Open `index.html` en voeg het volgende `<footer>`-element onder aan het `<aside>`-element toe:

```
<footer>
    <ul class="galerij">
        <li><a href="#"> </a></li>
        <li><a href="#"> </a></li>
        <li><a href="#"> </a></li>
        <li><a href="#"> </a></li>
    </ul>
    <br /><br /><br /><br />
    <small>&copy; Marathon van Amsterdam </small>
</footer>

</section> <!-- id="content"-->
</body>
</html>
```

1.13.2 Footer-galerij

In het `<footer>`-element hebben we een galerij gemaakt met een lijst van foto's met een hyperlink. Daar naast hebben we bedrijfsinformatie en copyrightinformatie toegevoegd.

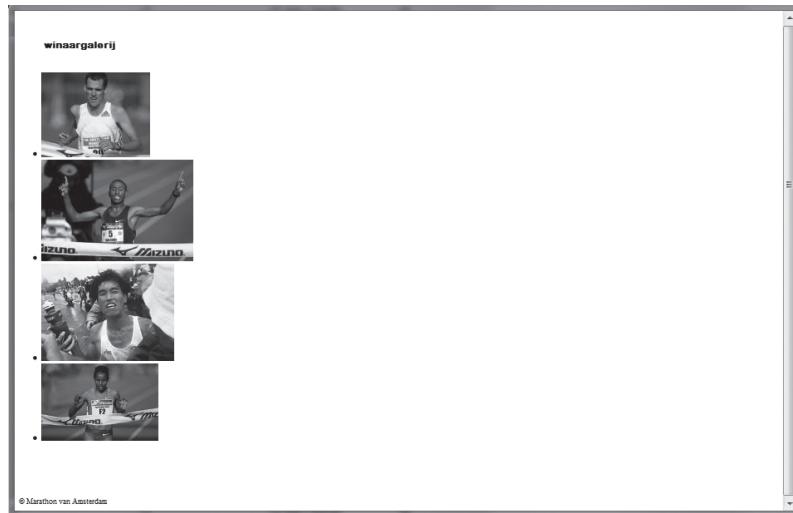
1.13.3 HTML Character Entity Encoding

Voor speciale symbolen zoals het euro- of het copyrightteken heeft HTML Character Entity Encoding ontwikkeld. We kunnen deze symbolen op twee manieren coderen: op naam of via numerieke encoding. Hieronder vind je een tabel met de meest voorkomende symbolen.

Tabel 1.4 Character Entity Encoding

Symbol	Encoding	Numeriek	Beschrijving
 	 	 	spatie
©	©	©	copyright
®	®	®	registreerde TM
™	™	™	trademark
€	€	€	euro
£	£	£	pound
¥	¥	¥	yen
&	&	&	ampersand
<	<	<	kleiner dan
>	>	>	groter dan
“	"	"	aanhalingstekens

Zoek je eigen finishfoto's en plaats ze in je map images. Het resultaat moet er zoals de volgende figuur uitzien:

Figuur 1.29
Galerij

1.13.4 Code camp – Lab 8

Open je eigen marathonproject en voeg een `<footer>`-element met vier foto's eraan toe. Maak je eigen footer.

1.14 Footer design

In deze paragraaf over design gaan we speciale effecten aan het `<footer>`-element toepassen. Ten eerste bepalen we de afmeting van het `<footer>`-element.

1.14.1 Afmeting van het element <footer>

In het volgende voorbeeld bepalen we de afmeting van het <footer>-element en geven we alle afbeeldingen in de footer dezelfde breedte.

Opgave 34 Open **styles.css** en voeg de volgende designattributen eraan toe:

```
footer{
    width: 750px;
    float: left;
    margin: 5px;
    padding: 5px;
    border: 1px solid #ddd;
}

footer ul li{
    float: left;
    padding: 10px;
    list-style: none;
}
footer ul li a img{
    width:100px;
    padding: 10px;
}
```

1.14.2 Het attribuut transform-origin

Het attribuut `transform-origin` wijzigt de beginpositie van een element. In het volgende voorbeeld worden de afbeeldingen geankerd en vergroot vanuit de hoek linksonder:

```
transform-origin:bottom left;
```

Opgave 35 Open **styles.css** en voeg de volgende designattributen eraan toe:

```
footer ul.galerij li a:hover img {
    -webkit-box-shadow: 4px 4px 10px rgba(0,0,0,0.5);
    -moz-box-shadow: 4px 4px 10px rgba(0,0,0,0.5);
    -o-box-shadow: 4px 4px 10px rgba(0,0,0,0.5);
    box-shadow: 4px 4px 10px rgba(0,0,0,0.5);

    -webkit-transform-origin:bottom left;
    -moz-transform-origin:bottom left;
    -o-transform-origin:bottom left;
    transform-origin:bottom left;

    -webkit-transform: scale(1.5);
    -moz-transform: scale(1.5);
    -o-transform: scale(1.5);
    transform: scale(1.5);

    -webkit-border-radius: 10px;
    -moz-border-radius: 10px;
    -o-border-radius: 10px;
    border-radius: 10px;
}
```

Het resultaat moet er zoals de volgende figuur uitzien.

Figuur 1.30

Positie van een element bepalen



1.14.3 Code camp – Lab 9

Open je eigen marathonproject en voeg de nieuwe designstyles aan het <footer>-element toe. Ontwerp je eigen footer.

1.14.4 CSS-animaties

In deze paragraaf gaan we een animatie voor het logo van de website coderen, zodat als je met de muis over het logo beweegt het rode logo naar rechts verplaatst het groene logo naar links verplaatst en het blauwe logo in een cirkel ronddraait.

Opgave 36 Open **styles.css** en voeg de volgende designattributen eraan toe:

```
#intro{
    -webkit-transition: all 6s ease-in;
    -moz-transition: all 6s ease-in;
    -o-transition: all 6s ease-in;
    -ms-transition: all 6s ease-in;
}

#rodelogo, #blauwelogo, #groenelogo{
    -webkit-transition: all 4s ease-in;
    -moz-transition: all 4s ease-in;
    -o-transition: all 4s ease-in;
    -ms-transition: all 4s ease-in;
}

#intro:hover #rodelogo{
    -webkit-transform: translate(100%,0px);
    -moz-transform: translate(100%,0px);
```

```

-o-transform: translate(100%,0px);
-ms-transform: translate(100%,0px);
}
#intro:hover #groenelogo{
-webkit-transform: translate(-100px,0px);
-moz-transform: translate(-100px,0px);
-o-transform: translate(-100px,0px);
-ms-transform: translate(-100px,0px);
}

#intro:hover #blauwelogo{
-webkit-transform: rotate(360deg);
-moz-transform: rotate(360deg);
-o-transform: rotate(360deg);
-ms-transform: rotate(360deg);
}

```

Het resultaat moet er als de volgende figuur uitzien:

Figuur 1.31
CSS-animaties



1.14.5 Code camp – Lab 10

In dit lab ga je de designstyles die je in deze paragraaf geoefend hebt toepassen op je webpagina uit Lab 9. Ontwerp je eigen animatie.

Een site is aantrekkelijk wanneer de content goed is vormgegeven. Een rich media site bevat afbeeldingen, geluid, video's en animaties. De site moet tegelijkertijd betrouwbaar zijn. In dit hoofdstuk gaan we verder met het ontwerpen van de navigatie van onze site.

2.1 Markup van de pagina Inschrijven

Nu we de homepage (ook wel startpagina) hebben ontworpen gaan we verder met de pagina Inschrijven. Hiervoor gebruiken we dezelfde layout als voor de homepagina. In deze paragraaf maken we het volgende formulier aan.

Figuur 2.1
Pagina Inschrijven
ontwerpen

Check een of meer sponsors
Voedselbank: Stichtingx: Stichtingz:

Ik loop mee als
Individu: Groep: Bedrijf:

Voer je gegevens in

E-mailadres	email	Mobiele telefoon	06	
Naam	naam	Adres	adres	Postcode
postcode	Plaats	Onderdeel		
Marathon	Ranking	Reactie		

Verzenden

Het enige wat we gaan veranderen is het `<nav>`-element en het `<article>`-element. In het `<nav>`-element wijzigen we de link naar de homepage en de link naar de pagina Inschrijven.

- Opgave** 1 Open **index.html** en sla het op als **inschrijven.html**. Wijzig het `<nav>`-element als volgt:

```
<nav>
  <ul>
    <li><a href="index.html">Home</a></li>
    <li><a href="#top">Inschrijven</a></li>
    <li><a href="nieuwsbrief.html">Nieuwsbrief</a>
    </li>
    <li><a href="resultaten.html">Resultaten</a>
    </li>
  </ul>
</nav>
```

2.1.1 Het element `<form>`

In het `<article>`-element geven we het `<h1>`-element binnen het `<header>`-element de nieuwe artikeltitel: Inschrijven. Daarna maken we een inschrijfformulier.

Om het formulier te openen maak je gebruik van de `<form>`-tag. Alles wat tussen `<form>` en `</form>` staat, zal herkend worden als onderdeel van het formulier.

Aan de `<form>`-tag kun je een actie (action) en een methode (method) toekennen. Zo kunnen de gegevens in een formulier in combinatie met scripts (bijvoorbeeld PHP) via de server lopen. Om de gegevens te kunnen mailen gebruik je bij action de term `mailto:`.

```
<form action ="mailto:info@mydomain.nl" method="post">
```

Bij method geef je aan wat er met de informatie gedaan moet worden. In dit voorbeeld kiezen we voor POST, want de informatie moet verzonden worden.

- Opgave** 2 Open **index.html** en sla het op als **inschrijven.html**. Wijzig het `<article>`-element als volgt:

```
<article class="rondehoeken">
  <header>
    <div class="streep1"></div>
    <div class="streep2"></div>
    <div class="streep3"></div>
    <div class="streep4"></div>
    <div class="streep5"></div>
    <h1>Inschrijven</h1>
  </header>
  <div id="artikel-container">
```

```
<div id="form-container">
<form class="rondehoeken" name="inschrijffformulier"
      action="info@adres.com" method="post">
</form>

</div> <!-- id="form-container"-->
</div> <!-- id="artikel-container" -->
</article>
```

2.1.2 Het element <fieldset>

Het <fieldset>-element mag een aantal van de volgende elementen bevatten:

- invoervelden
- opsommingstekens (buttons)
- selectievakjes (checkboxes)
- keuzerondjes (radio buttons)
- keuzemenu (dropdown-menu)

2.1.3 Het element <legend>

Het <legend>-element is de legenda voor een groep formulierelementen.

2.1.4 Het element <label>

Het <label>-element is de label of beschrijving van een formulier-element.

2.1.5 Het element <input type="checkbox">

Selectievakjes (checkboxes) gebruik je om een of meer opties te kunnen kiezen. Een selectievak maak je op de volgende manier:

Opgave 3 Open **inschrijven.html** en voeg het volgende <fieldset>-element eraan toe.

```
<fieldset>
<legend>Check een of meer sponsors</legend>
Voedselbank:
<input type="checkbox" name="voedselbank" id="voedselbank"
value="v" />
Stichtingx:
<input type="checkbox" name="stichtingx" id="stichtingx"
value="x" />
Stichtingz:
<input type="checkbox" name="stichtingz" id="stichtingz"
value="z" />
</fieldset>
```

2.1.6 Het element <input type="radio">

Een keuzerondje (radio button) gebruik je uitsluitend een optie uit een lijst te kunnen kiezen. Een keuzerondje maak je op de volgende manier:

- Opgave 4** Open **inschrijven.html** en voeg het volgende **<fieldset>**-element eraan toe.

```
<fieldset>
<legend>Ik loop mee als</legend>
  Individu:
<input type="radio" name="deelnemer" id="individu" value="i" />
  Groep:
<input type="radio" name="deelnemer" id="groep" value="g" />
  Bedrijf:
<input type="radio" name="deelnemer" id="bedrijf" value="b" />
</fieldset>
```

Bij `input type="radio"` geef je de hele groep dezelfde naam bij voorbeeld:

`name="deelnemer"`

en er kan maar één keuzerondje geselecteerd worden.

2.1.7 Het element `<input type="text">`

Het `<input type= "text" >`-element is een formulierelement met een invulvenster.

Bijvoorbeeld:

```
<fieldset>
<legend>Inschrijfformulier</legend>
<label for="voornaam">Uw voornaam: </label>
<input required type="text" id="voornaam"
placeholder="vul uw voornaam in" />
</fieldset>
```

- `input type="text"` geeft aan dat het een tekstinvulvenster is.
- `id="voornaam"` geeft aan dat de referentienaam van het veld `voornaam` is.
- `required` geeft aan dat het invulvenster verplicht ingevuld moet worden.
- `placeholder` geeft aan dat er een grijze tekst is die automatisch in het invulvenster verschijnt.

- Opgave 5** Open **inschrijven.html** en voeg het volgende **<fieldset>**-element eraan toe.

```
<fieldset>
<legend>Voer je gegevens in</legend>
<label for="gebruiker">E-mailadres</label>
<input required type="email" id="email" name="email"
placeholder="email" />
<label for="mobiele telefoon">Mobiele telefoon</label>
<input required type="text" id="mobiele telefoon"
name="mobiele telefoon" />
```

```
placeholder="06" />
<label for="naam">Naam</label>
<input required type="text" id="naam" name="naam"
placeholder="naam" />
<label for="adres">Adres</label>
<input required type="text" id="adres" name="adres"
placeholder="adres" />
<label for="postcode">Postcode</label>
<input required type="text" id="postcode" name="postcode"
placeholder="postcode" />

<label for="plaats">Plaats</label>
<input type="text" name="plaats" id="plaats" list="steden">
<datalist id="steden">
<option value="Almere">
<option value="Amstelveen">
<option value="Amsterdam">
<option value="Den Haag">
<option value="Hilversum">
<option value="Leeuwarden">
<option value="Maastricht">
<option value="Naarden">
<option value="Utrecht">
</datalist>

<label for="onderdeel">Onderdeel</label>
<select name="onderdeel">
<option value="marathon">Marathon</option>
<option value="halve">Halve marathon</option>
<option value="8">8 km</option>
<option value="kids">Olympic kids 1km</option>
</select>

<label for="ranking">Ranking</label>
<input type="range" name="ranking" id="ranking" min="1"
max="5">
<label for="reactie">Reactie</label>
<textarea cols="35" rows="4" name="reactie" id="reactie"
placeholder="Mail ons je reactie">
</textarea>
<br />
<input type="submit" id="submit" name="submit"
value="Verzenden" />
</fieldset>
```

2.1.8 Het element <select>

Om meerdere keuzemogelijkheden te bieden kun je gebruikmaken van een keuzemenu (dropdown-menu). Ook hier geef je bij `value` aan voor welke keuzemogelijkheid gekozen wordt. `Name` gebruik je weer voor een algemene beschrijving van de groep. Bijvoorbeeld, in de vorige opgave hebben we het volgende gecodeerd:

```
<label for="onderdeel">Onderdeel</label>
<select name="onderdeel">
<option value="marathon">Marathon</option>
<option value="halve">Halve marathon</option>
<option value="8">8 km</option>
<option value="kids">Olympic kids 1km</option>
</select>
```

Bij een `<select>`-element heb je alleen maar de keuze uit de geboden opties.

2.1.9 Het element `<datalist>`

Het `<datalist>`-element is een combinatie van een `input type="text"` en een dropdown-menu. Bijvoorbeeld, in de vorige opgave hebben we het volgende gecodeerd:

```
<label for "plaats">Plaats</label>
<input type="text" name="plaats" id="plaats" list="steden">
<datalist id="steden">
<option value="Amstelveen">
<option value="Amsterdam">
<option value="Hilversum">
</datalist>
```

Bij een `<datalist>`-element mag je zelf een niet beschikbare optie invullen. In dit geval hoort Zwolle niet tot de gegeven opties maar je mag Zwolle wel intypen.

2.1.10 Het element `<input type="range">`

Bij het `<input type="range">`-element mag je een getal tussen een minimale en maximale waarde invoeren.

Bijvoorbeeld, in de vorige opgave hebben we het volgende gecodeerd:

```
<label for="ranking">Ranking</label>
<input type="range" name="ranking" id="ranking" min="1"
max="5">
```

2.1.11 Het element `<textarea>`

Omdat een tekstveld weinig ruimte biedt om veel tekst kwijt te kunnen, kun je ook gebruikmaken van het `<textarea>`-element.

Bijvoorbeeld, in de vorige opgave hebben we het volgende gecodeerd:

```
<label for="reactie">Reactie</label>
<textarea cols="35" rows="4" name="reactie" id="reactie"
placeholder="mail ons je reactie">
</textarea>
```

2.1.12 Het element <input type="submit">

Bij onlineformulieren kun je ook knoppen (buttons) gebruiken. Een verplicht onderdeel van een formulier is de verzendknop (submit button). Bijvoorbeeld, in de vorige opgave hebben we het volgende gecodeerd:

```
<input type="submit" id="submit" name="submit"
value="Verzenden" />
```

Het resultaat van de vorige opgave zie je in de volgende figuur.

Figuur 2.2
Formulier maken

2.1.13 Code camp – Lab 11

Open je eigen marathonproject en voeg een pagina **Inschrijven.html** eraan toe. Het resultaat moet er ongeveer zoals de vorige figuur uitzien.

2.2 Design van de pagina Inschrijven

Voor het design van de inschrijfpagina maken we gebruik van de volgende nieuwe CSS3 attributen.

- Clear
- After
- Display
- Required
- Focus
- Invalid
- Radial-gradient

In deze paragraaf maken we het design van het volgende formulier aan.

Figuur 2.3
Formulier
ontwerpen

The screenshot shows a web-based form titled "Inschrijven" (Register) for a marathon. At the top, there are three decorative icons with the text "I am amsterdam". The form has several sections:

- Check een of meer sponsors**: Options for "Voedselbank", "Stichtingx", and "Stichtingz".
- Ik loop mee als**: Options for "Individu", "Groep", and "Bedrijf".
- Voer je gegevens in**:

E-mailadres:	<input type="text"/> email
Mobiele telefoon:	<input type="text"/> 06
Naam:	<input type="text"/> naam
Adres:	<input type="text"/> adres
Postcode:	<input type="text"/> postcode
Plaats:	<input type="text"/>
Onderdeel:	<input type="button" value="Marathon"/>
Ranking:	<input type="text"/>
Reactie:	<input type="text"/>
- Verzenden** button at the bottom.

Opgave 6 Open styles.css en voeg de volgende designattributen eraan toe:

```

form {
    border: solid 2px #888;
}

fieldset{
    margin:8px 0;
    padding:8px;
    font-family: LeagueGothic, Tahoma, Geneva, sans-serif;
    font-size: 16px;
}

label{
    float:left;
    clear:left;
    padding-top:2px;
    width:120px;
    text-align:right;
}

label:after{
    content:":"
}

label strong{
    color:#c00
}

```

2.2.1 Formulierdesign

In bovenstaande opgave heeft het `<form>`-element een solide grijze border gekregen:

```
form {  
    border: solid 2px #888;  
}
```

Alle `<fieldset>`-elementen hebben hetzelfde attribuut font-family gekregen:

```
fieldset{  
    margin:8px 0;  
    padding:8px;  
    font-family: LeagueGothic, Tahoma, Geneva, sans-serif;  
    font-size: 16px;  
}
```

2.2.2 Het attribuut clear:

Elk `<label>`-element krijgt een attribuut float met als waarde left en daarna een attribuut clear met als waarde left. Zo zorg je ervoor dat het attribuut float:left wordt gereset voor de volgende elementen:

```
label{  
    float:left;  
    clear:left;  
    padding-top:2px;  
    width:120px;  
    text-align:right;  
}
```

2.2.3 Het attribuut after:

Alle `<label>`-elementen krijgen een attribuut after. Met dit attribuut kun je meer content, in dit geval ":" aan een element toevoegen. Je ziet dat alle labels eindigen met een ():-teken.

```
label:after{  
    content:":"  
}
```

2.2.4 Het attribuut display:

Het attribuut display geeft aan hoe een element weergegeven wordt. De opties zijn:

Een attribuut `display:block` neemt alle mogelijke ruimte en krijgt een `
` voor en na het element.

Een attribuut `display:none` wordt niet weergegeven en neemt ook geen ruimte in.

In de volgende opgave zorgen we ervoor dat alle `select`, `textarea` en `<input>`-elementen een attribuut `display:block` en ronde hoeken van 5px krijgen.

Opgaven 7 Open **styles.css** en voeg de volgende designattributen eraan toe:

```
input[type=text], input[type=email], input[type=submit],
input[type=range], select, textarea{

    border: solid 1px rgb(0, 0, 0);
    display:block;
    margin: 10px auto 10px;
    padding: 5px 30px 5px 50px;
    width: 50%;
    text-align:center;
    font-size: 16px;

    -webkit-border-radius:5px;
    -moz-border-radius:5px;
    -o-border-radius:5px;
    border-radius:5px;
}
```

8 Open **styles.css** en voeg de volgende designattributen eraan toe:

```
input:not([type=range]):not([type=date]):not([type=submit])
:not([type=button]):not([type=checkbox]):not([type=number])
:required {
    background-image: url('images/required.png');
    background-repeat: no-repeat;
}

input:focus{
    -webkit-box-shadow: 4px 4px 10px rgba(254,225,0, 0.5);
    -moz-box-shadow: 4px 4px 10px rgba(254,225,0, 0.5);
    -o-box-shadow: 4px 4px 10px rgba(254,225,0, 0.5);
    box-shadow: 4px 4px 10px rgba(254,225,0, 0.5);
}

invalid {
    -webkit-box-shadow: 4px 4px 10px rgba(193, 28, 54, 0.5);
    -moz-box-shadow: 4px 4px 10px rgba(193, 28, 54, 0.5);
    -o-box-shadow: 4px 4px 10px rgba(193, 28, 54, 0.5);
    box-shadow: 4px 4px 10px rgba(193, 28, 54, 0.5);
}
```

2.2.5 Het attribuut required:

Het attribuut required geeft aan dat een invulvenster verplicht moet worden ingevuld. In bovenstaande opgave krijgen alle `<input type="text">`-elementen met een attribuut required dezelfde achtergrondafbeelding:

Figuur 2.4
Het attribuut
required



```
input:not([type=range]):not([type=date]):not([type=submit])
:not([type=button]):not([type=checkbox]):not([type=number])
:required {
    background-image: url('images/required.png');
    background-repeat: no-repeat;
}
```

2.2.6 Het attribuut focus:

Een `<input type="text">`-element is gefocust wanneer het element wordt ingevuld. In dit geval krijgt een gefocust element en gele box-shadow:

```
input:focus{
    -webkit-box-shadow: 4px 4px 10px rgba(254,225,0, 0.5);
    -moz-box-shadow: 4px 4px 10px rgba(254,225,0, 0.5);
    -o-box-shadow: 4px 4px 10px rgba(254,225,0, 0.5);
    box-shadow: 4px 4px 10px rgba(254,225,0, 0.5);
}
```

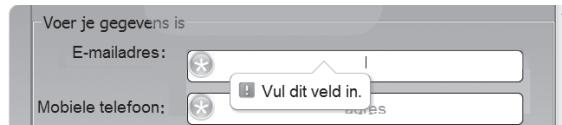
2.2.7 Het attribuut invalid:

Met het attribuut `invalid` kun je controleren of de ingevoerde input geldig is. Bijvoorbeeld, een element dat ‘required’ moet niet leeg zijn of een e-mailadres moet de juiste vorm hebben. In dit geval krijgt een element met ongeldig input en rode box-shadow:

```
invalid {
    -webkit-box-shadow: 4px 4px 10px rgba(193, 28, 54, 0.5);
    -moz-box-shadow: 4px 4px 10px rgba(193, 28, 54, 0.5);
    -o-box-shadow: 4px 4px 10px rgba(193, 28, 54, 0.5);
    box-shadow: 4px 4px 10px rgba(193, 28, 54, 0.5);
}
```

Op het moment dat je op de knop Verzenden klikt, geeft een element dat verplicht is maar geen input bevat onderstaande melding: Vul dit veld in.

Figuur 2.5
Element zonder
input



Op het moment dat je op Verzenden klikt, geeft een e-mail-element met een ongeldig e-mailadres onderstaande melding: Voer een e-mailadres in. De lege verplichte elementen krijgen een rode box-shadow.

Figuur 2.6
Verplichte
elementen



2.2.8 De function radial-gradient()

Een radial-arcering begint in het centrum en loopt naar de randen van het element. De syntax is als volgt:

```
radial-gradient(middenpunt -x middenpunt -y, circle of
ellipse, beginkleur, eindkleur)
```

Bijvoorbeeld:

```
-radial-gradient(50% 50%, ellipse,
    rgba(82, 172, 149,1) 0%,
    rgba(199, 235, 210,1) 100%);
```

Opgave 9 Open **styles.css** en voeg de volgende designattributen eraan toe:

```
input[type=submit] {
    background: -moz-radial-gradient(50% 50%, ellipse,
        rgba(82, 172, 149,1) 0%,
        rgba(199, 235, 210,1) 100%);
    background: -webkit-radial-gradient(50% 50%, ellipse,
        rgba(82, 172, 149,1) 0%,
        rgba(199, 235, 210,1) 100%);
    background: radial-gradient(50% 50%, ellipse,
        rgba(82, 172, 149,1) 0%,
        rgba(199, 235, 210,1) 100%);
    width: 100%;
    opacity: 0.6;
}

input[type=submit]:hover{
    opacity: 1;
    -webkit-transition: opacity 0.5s linear;
    -moz-transition: opacity 0.5s linear;
    -o-transition: opacity 0.5s linear;
    transition: opacity 0.5s linear;
}
```

Het resultaat van de vorige opgave zie je in de volgende figuur. Je ziet ook het hovereffect op de verzendknop.

Figuur 2.7
Hovereffect

The screenshot shows a web page for the 'Amsterdam Marathon'. At the top, there's a navigation bar with links for 'Home', 'Inschrijven', 'Nieuwsbrief', and 'Resultaten'. Below the navigation, there's a large input field containing the text 'I amsterdam'. The main content area contains a registration form with the following fields:
 - E-mailadres: An input field with a placeholder 'email' and a small icon.
 - Mobiele telefoon: An input field with a placeholder '06' and a small icon.
 - Naam: An input field with a placeholder 'naam' and a small icon.
 - Adres: An input field with a placeholder 'adres' and a small icon.
 - Postcode: An input field with a placeholder 'postcode' and a small icon.
 - Plaats: An input field with a placeholder and a small icon.
 - Onderdeel: A dropdown menu set to 'Marathon'.
 - Ranking: A slider control.
 - Reactie: A text area for comments.
 At the bottom of the form is a large, rounded rectangular button labeled 'Verzenden' in white text. To the right of the form, there's a sidebar with a video player showing a race, a map of the course, and a logo for 'TUNIC CHOCOLONEE'.

2.2.9 Code camp – Lab 12

In dit lab ga je de designstyles die je in deze paragraaf geoefend hebt toe-passen op je webpagina uit Lab 11. Ontwerp je eigen verzendknop.

2.3 Markup van de pagina Nieuwsbrief

We gaan nu de lay-out van de pagina Nieuwsbrief maken. In deze pagina gebruiken we dezelfde lay-out als van de startpagina. Het resultaat zie je in de volgende figuur:

Figuur 2.8
Pagina Nieuwsbrief



Het enige wat we gaan veranderen is het `<nav>`-element en het `<article>`-element. In het `<nav>`-element wijzigen we de link naar de inschrijfpagina en de link naar de nieuwsbriefpagina.

Opgave 10 Open `inschrijven.html` en sla het op als `nieuwsbrief.html`. Wijzig het `<nav>`-element als volgt:

```
<nav>
  <ul>
    <li><a href="index.html">Home</a></li>
    <li><a href="inschrijven.html">Inschrijven</a></li>
    <li><a href="#top">Nieuwsbrief</a></li>
    <li><a href="resultaten.html">Resultaten</a></li>
  </ul>
</nav>
```

2.3.1 Het element <iframe>

Het <iframe>-element maakt een inline-frame zodat je een tweede document kan inbedden binnen het huidige document. In de volgende opgave gaan we een inline-frame coderen voor een YouTube-filmpje over de Amsterdam Marathon.

- Opgave 11** Open **nieuwsbrief.html**. Vervang het huidige <article>-element met het volgende <article>-element:

```
<article class="rondehoeken">
<header>
    <div class="streep1"></div>
    <div class="streep2"></div>
    <div class="streep3"></div>
    <div class="streep4"></div>
    <div class="streep5"></div>
    <h1 id="artikel-titel" >Nieuwsbrief</h1>
</header>
<div id="artikel-container">
    <iframe id="youtube-iframe" class="youtube-player"
        type="text/html" src="http://www.youtube.com/embed/TouQIhP-
        pQY" ></iframe>
    <h2>Mauris vel neque</h2>
    <div id="item">
        
        <p>Aliquam erat volutpat. Mauris vel neque sit amet nunc
            gravida congue sed sit amet purus. Quisque lacus quam,
            egestas ac tincidunt a, lacinia vel velit. Morbi ac commodo
            nulla.</p>
    </div><!-- id="item"-->
    <h2>Aliquam erat volutpat</h2>
    <div id="item">
        
        <p>Aliquam erat volutpat. Mauris vel neque sit amet nunc
            gravida congue sed sit amet purus. Quisque lacus quam,
            egestas ac tincidunt a, lacinia vel velit. Morbi ac commodo
            nulla.</p>
    </div><!-- id="item"-->
    <h2>Quisque lacus quam</h2>
    <div id="item">
        
        <p>Aliquam erat volutpat. Mauris vel neque sit amet nunc
            gravida congue sed sit amet purus. Quisque lacus quam,
            egestas ac tincidunt a, lacinia vel velit. Morbi ac commodo
            nulla.</p>
    </div><!-- id="item"-->
```

```
</div> <!-- id="nieuw-artikel-container"-->  
</article>
```

2.3.2 Code camp – Lab 13

Open je eigen marathonproject en voeg een pagina **nieuwsbrief.html** eraan toe. Ontwerp je eigen nieuwsbrief.

2.4 Design van de pagina Nieuwsbrief

Voor het design van de pagina Nieuwsbrief hoeven we alleen maar de designattributen voor het `<iframe>`-element te definiëren.

2.4.1 Het attribuut frameborder

Het attribuut frameborder geeft aan of het `<iframe>`-element een border krijgt.

Opgave 12 Open **styles.css** en voeg de volgende designattributen eraan toe:

```
#youtube-iframe{  
    width:490px;  
    height:200px;  
    frameborder: 0;  
}
```

2.4.2 Code camp – Lab 14

In dit lab ga je de designstyles die je in deze paragraaf geoefend hebt toepassen op je webpagina uit Lab 13. Ontwerp je eigen afmetingen en frameborders voor je YouTube-film.

2.5 Markup van de Resultatenpagina

We gaan nu de lay-out van de Resultatenpagina maken. In deze pagina gebruiken we dezelfde lay-out als van de startpagina. Het resultaat zie je in de volgende figuur.

Figuur 2.9
Resultatenpagina

Finish	Naam	Tijd	Record	Land
1	Gebre Gebremariam	02:08:14		Ethiopia
2	Emmanuel Mutai	02:09:18		Kenya
3	Moses Kigen Kipkosgei	02:10:39		Kenya
4	Abderrahim Goumri	02:10:51		Morocco
5	James Kwambai	02:11:31		Kenya
6	Meb Keflezighi	02:11:38		VS
7	Marilson Gomes Dos Santos	02:11:51		Brazil
8	Dathan Ritzenhein	02:12:33		VS
9	Abel Kirui	02:13:01		Kenya
10	Abderrahime Bouram dane	02:14:07		Morocco

Het enige wat we gaan veranderen is het `<nav>`-element en het `<article>`-element. In het `<nav>`-element wijzigen we de link naar de nieuwsbriefpagina en de link naar de resultatenpagina.

Opgave 13 Open **nieuwsbrief.html** en sla het op als **resultaten.html**. Wijzig het `<nav>`-element als volgt:

```
<nav>
  <ul>
    <li><a href="index.html">Home</a></li>
    <li><a href="inschrijven.html">Inschrijven</a></li>
    <li><a href="nieuwsbrief.html">Nieuwsbrief</a></li>
    <li><a href="#top">Resultaten</a></li>
  </ul>
</nav>
```

2.5.1 Het element `<table>`

Een tabel coderen we met het `<table>`-element. De eenvoudigste tabel bestaat uit één cel. Je mag een tabel splitsen in drie secties: een koptekst (header), een voettekst (footer) en de tabel zelf (body).

2.5.2 Het element `<thead>`

Het `<thead>`-element gebruiken we om alle `<th>`-elementen of alle kopteksten van een tabel te groeperen. Bijvoorbeeld:

```
<table class="table1">
  <thead>
    <tr>
      <th>Finish</th>
      <th>Naam</th>
      <th>Tijd</th>
      <th>Record</th>
      <th>Land</th>
    </tr>
  </thead>
</table>
```

2.5.3 Het element `<tr>`

Het `<tr>`-element markeert het begin van een tabelrij.

2.5.4 Het element `<th>`

Het `<th>`-element gebruiken we voor kopteksten van de `<thead>`-elementen en voor de voetteksten van de `<tfoot>`-elementen.

2.5.5 Het element `<tfoot>`

Het `<tfoot>`-element gebruiken we om alle voetteksten van een tabel te groeperen. Bijvoorbeeld:

```
<tfoot>
<tr>
  <th colspan="5">Top 10 Rankings</th>
</tr>
</tfoot>
```

2.5.6 Het attribuut colspan

Het attribuut colspan geeft de breedte in kolommen weer van een element. In dit geval is de voettekst van de tabel 5 kolommen breed.

Figuur 2.10

Kolommen in een tabel

Finish	Naam	Tijd	Record	Land
Top 10 Rankings				

2.5.7 Het element <tbody>

Het `<tbody>`-element gebruiken we om alle `<td>`-elementen of alle cellen van een tabel te groeperen. Bijvoorbeeld:

```
<tbody>
<tr>
  <td>1</td>
  <td>Gebre Gebrmariam</td>
  <td>02:08:14</td>
  <td><span class="check"></span></td>
  <td>Ethiopia</td>
</tr>
</tbody>
```

2.5.8 Het element <td>

Met het `<td>`-element kunnen we een tabelrij in cellen splitsen. In dit geval splitsen we alle rijen van het `<tbody>`-element in 5 cellen.

Figuur 2.11

Cellen

Finish	Naam	Tijd	Record	Land
1	Gebre Gebrmariam	02:08:14		Ethiopia
2	Emmanuel Mutai	02:09:18		Kenya
Top 10 Rankings				

Opgave 14 Open `resultaten.html`. Vervang het `<article>`-element door het volgende `<article>`-element:

```
<article class="rondehoeken">
<header>
  <div class="streep1"></div>
  <div class="streep2"></div>
  <div class="streep3"></div>
  <div class="streep4"></div>
  <div class="streep5"></div>
  <h1 id="artikel-titel" >Top 10 runners</h1>
</header>
<div id="artikel-container">
  <table class="table1">
    <thead>
      <tr>
        <th>Finish</th>
        <th>Naam</th>
        <th>Tijd</th>
        <th>Record</th>
        <th>Land</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>1</td>
        <td>Gebre Gebrmariam</td>
        <td>02:08:14</td>
        <td><span class="check"></span></td>
        <td>Ethiopia</td>
      </tr>
      <tr>
        <td>2</td>
        <td>Emmanuel Mutai</td>
        <td>02:09:18</td>
        <td><span class="check"></span></td>
        <td>Kenya</td>
      </tr>
    </tbody>
  </table>
</div>
```

```
</thead>
<tfoot>
<tr>
    <th colspan="5">Top 10 Rankings</th>
</tr>
</tfoot>
<tbody>
<tr>
    <td>1</td>
    <td>Gebre Gebrmariam</td>
    <td>02:08:14</td>
    <td><span class="check"></span></td>
    <td>Ethiopia</td>
</tr>
<tr>
    <td>2</td>
    <td>Emmanuel Mutai</td>
    <td>02:09:18</td>
    <td> </td>
    <td>Kenya</td>
</tr>
<tr>
    <td>3</td>
    <td>Moses Kigen Kipkosgei</td>
    <td>02:10:39</td>
    <td> </td>
    <td>Kenya</td>
</tr>
<tr>
    <td>4</td>
    <td>Abderrahim Goumri</td>
    <td>02:10:51</td>
    <td><span class="check"></span></td>
    <td>Morocco</td>
</tr>
<tr>
    <td>5</td>
    <td>James Kwambai</td>
    <td>02:11:31</td>
    <td><span class="check"></span></td>
    <td>Kenya</td>
</tr>
<tr>
    <td>6</td>
    <td>Meb Keflezighi</td>
    <td>02:11:38</td>
    <td> </td>
    <td>VS</td>
</tr>
<tr>
    <td>7</td>
    <td>Marilson Gomes Dos Santos</td>
    <td>02:11:51</td>
```

```

<td><span class="check"></span></td>
<td>Brazil</td>
</tr>
<tr>
    <td>8</td>
    <td>Dathan Ritzenhein</td>
    <td>02:12:33</td>
    <td><span class="check"></span></td>
    <td>VS</td>
</tr>
<tr>
    <td>9</td>
    <td>Abel Kirui</td>
    <td>02:13:01</td>
    <td><span class="check"></span></td>
    <td>Kenya</td>
</tr>
<tr>
    <td>10</td>
    <td>Abderrahime Bouramdane</td>
    <td>02:14:07</td>
    <td><span class="check"></span></td>
    <td>Morocco</td>
</tr>
</tbody>
</table>
</div> <!-- id="nieuw-artikel-container" -->
</article>
```

Het resultaat van vorige opgave moet er zoals de volgende figuur uitzien:

Figuur 2.12
Resultatenlijst

The screenshot shows the results page for the Amsterdam Marathon. At the top, there's a navigation bar with links for Home, Inschrijven, Nieuwsbrief, Resultaten, and a button for I Amsterdam. Below the navigation, there's a banner with the text 'I Amsterdam' repeated three times. The main content area has a title 'Top 10 runners'. A table lists the top 10 finishers with their names, times, records, and countries. To the right of the table, there are three small images: a group of runners, a runner crossing a finish line, and a person holding a trophy. At the bottom, there's a footer with the text '© Marathon van Amsterdam'.

Finish	Naam	Tijd	Record	Land
1	Gebre Gebrmariam	02:08:14		Ethiopia
2	Emmanuel Mutai	02:09:18		Kenya
3	Moses Kigen Kipkosgei	02:10:39		Kenya
4	Abderrahim Gouuri	02:10:51		Morocco
5	James Kwambwa	02:11:31		Kenya
6	Meb Keflezighi	02:11:38		VS
7	Marlon Gomes Dos Santos	02:11:51		Brazil
8	Dathan Ritzenhein	02:12:33		VS
9	Abel Kirui	02:13:01		Kenya
10	Abderrahime Bouramdane	02:14:07		Morocco

2.5.9 Code camp – Lab 15

Open je eigen marathonproject en voeg een pagina **resultaten.html** eraan toe. Voeg een nieuwe tabelkolom eraan toe.

2.6 Design van de Resultatenpagina

Voor het design van de Resultatenpagina hoeven we alleen maar de designattributen voor alle `<table>`-elementen te definiëren. Het resultaat moet er als de volgende figuur uitzien:

Figuur 2.13

Ontwerp
Resultatenpagina

Finish	Naam	Tijd	Record	Land
1	Gebre Gebrmariam	02:08:14	✓	Ethiopia
2	Emmanuel Mutai	02:09:18		Kenya
3	Moses Kigen Kipkosgei	02:10:39		Kenya
4	Abderrahim Goumri	02:10:51	✓	Morocco
5	James Kwambai	02:11:31	✓	Kenya
6	Meb Keflezighi	02:11:38		VS
7	Marilson Gomes Dos Santos	02:11:51	✓	Brazil

2.6.1 Het attribuut border-collapse

Een tabelborder kan collapse (enkel) zijn of separate (aparte) zijn.
Bijvoorbeeld:

```
table{
    border-collapse:separate;
}
```

Bij een separate border krijgt iedere tabelcel een eigen (aparte) border.

Opgave 15 Open **styles.css** en voeg de volgende designattributen eraan toe:

```
table.table1{
    font-family: "Trebuchet MS", sans-serif;
    font-size: 14px;
    font-weight: bold;
    line-height: 1.0em;
    font-style: normal;
    border-collapse:separate;
}
.table1 thead th{
    padding:15px;
    color:#ffff;
    text-shadow:1px 1px 1px #568F23;
    border:1px solid #93CE37;
    border-bottom:3px solid #9ED929;
```

```
background-color:#9DD929;
background-color:rgb(207, 209, 242);

background:-webkit-gradient(
    linear,
    left bottom,
    left top,
    color-stop(0.02, rgb(86, 122, 199)),
    color-stop(0.51, rgb(148, 186, 231)),
    color-stop(0.87, rgb(233, 240, 250))
);
background: -moz-linear-gradient(
    center bottom,
    rgb(86, 122, 199) 2%,
    rgb(148, 186, 231) 51%,
    rgb(233, 240, 250) 87%
);

-webkit-border-top-left-radius:5px;
-webkit-border-top-right-radius:5px;
-moz-border-radius:5px 5px 0px 0px;
border-top-left-radius:5px;
border-top-right-radius:5px;
}

.table1 thead th:empty{
    background:transparent;
    border:none;
}
.table1 tfoot th{
    color:#666;
}
.table1 tbody td{
    padding:5px;
    text-align:center;
    background-color:#DEF3CA;
    border: 2px solid #E7EFE0;
    -moz-border-radius:2px;
    -webkit-border-radius:2px;
    border-radius:2px;
    color:#666;
    text-shadow:1px 1px 1px #fff;
}
.table1 tbody span.check::before{
    content : url(images/check0.png)
}
```

Sommige cellen van de kolom Record zijn leeg gebleven:

```
<td></td>
```

Andere cellen van de kolom Record hebben de volgende markup gekregen:

```
<td><span class="check"></span></td>
```

In de vorige opgave heeft de code:

```
.table1 tbody span.check::before{
    content : url(images/check0.png)
}
```

ervoor gezorgd dat de cellen met de `class="check"` de volgende figuur als content krijgen:

Figuur 2.14

Klasse



Het resultaat van de vorige opgaven ziet er als volgt uit:

Figuur 2.15

Definitieve versie
Resultatenpagina

Finish	Naam	Tijd	Record	Land
1	Gebre Gebrmariam	02:08:14	✓	Ethiopia
2	Emmanuel Mutai	02:09:18		Kenya
3	Moses Kigen Kipkosgei	02:10:39		Kenya
4	Abderrahim Goumri	02:10:51	✓	Morocco
5	James Kwambai	02:11:31	✓	Kenya
6	Meb Keflezighi	02:11:38		US
7	Marilson Gomes Dos Santos	02:11:51	✓	Brazil

2.6.2 Code camp – Lab 16

In dit lab ga je de designstyles die je in deze paragraaf geoefend hebt toe-passen op je webpagina uit Lab 15. Ontwerp je eigen kleurschema.

Maak nu Portfolio-opdracht 1 – Ontwerp webshop

3

JavaScript

3.1 Inleiding JavaScript

In dit deel 3 behandelen we basiskennis van JavaScript.

Objectgeoriënteerde JavaScript en jQuery worden behandeld in het boek *Webdesign voor mobiele applicaties*.

Een interactieve website is veel meer dan webpagina's met afbeeldingen. Een interactieve website kan vragen aan de gebruiker stellen en reacties krijgen. Om een site interactief te maken gebruik je JavaScript.

3.1.1 Wat is JavaScript?

JavaScript is in 1995 ontwikkeld door Brendan Eich van Netscape Communications Corporation speciaal voor het gebruik in Netscape Navigator. De oorspronkelijke naam was Mocha en later LiveScript. De naam JavaScript komt uit de tijd dat in de Netscape-browser ook ondersteuning voor Java-applets werd ingebouwd. Door het gebruik van JavaScript werd het mogelijk om webpagina's interactief te maken. Dynamic HTML was hier een logisch vervolg op.

JScript is de Microsoft-implementatie van JavaScript. JavaScript is een scripttaal. Een script is een stukje code op een HTML-pagina. Met JavaScript voeg je functionaliteit aan je webpagina's toe. Er bestaat client-side en server-side JavaScript.

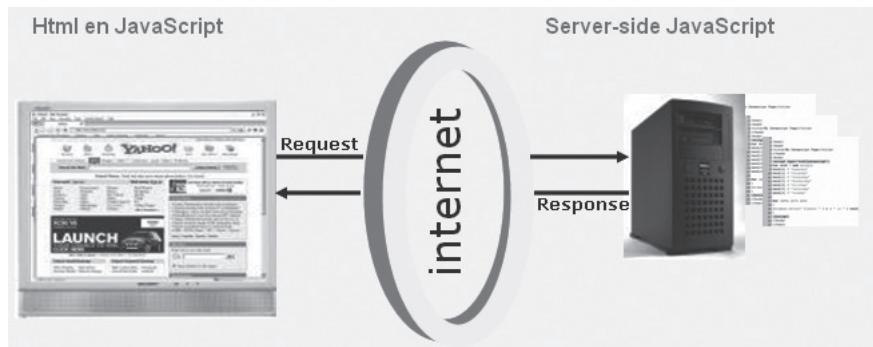
- Client-side JavaScript wordt direct uitgevoerd via de browser van de bezoeker (client).
- Server-side JavaScript wordt uitgevoerd via een server, waarna het resultaat wordt teruggestuurd naar de browser van de bezoeker.

De browser herkent een JavaScript tussen de volgende `<script>`-tags. De syntaxis is als volgt:

```
<script type="text/javascript">
...
</script>
```

Figuur 3.1

Client-side en server-side JavaScript



Op de puntjes coderen we JavaScript-code. Alle JavaScript-code tussen de `<script>`-tags vormt de body van je script. In de volgende opgave coderen we ons eerste JavaScript.

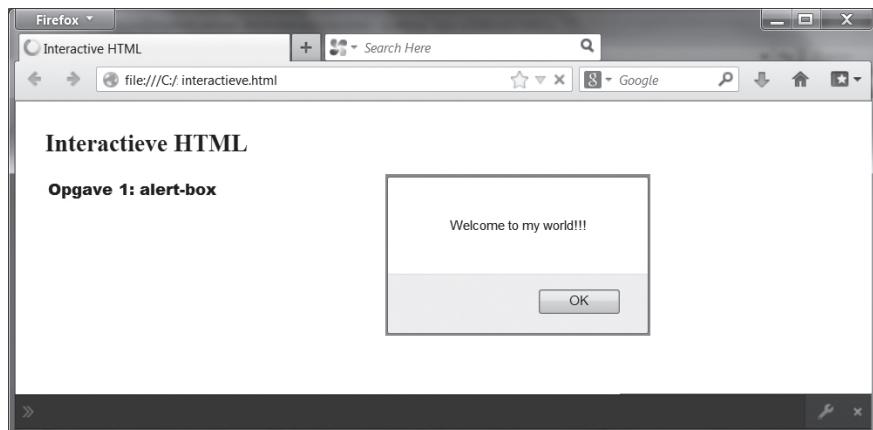
Opgave

- Codeer het volgende script met je favoriete HTML-editor en sla het op als **interactieve.html**

```
<!DOCTYPE html>
<html lang="nl">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
            charset=UTF-8">
        <title>Interactive HTML</title>
    </head>
    <body>
        <h3>Interactive HTML</h3>
        <p>Opgave 1: alertbox</p>
        <script type="text/javascript">
            alert("Welcome to my world!!!");
        </script>
    </body>
</html>
```

- Het woord `alert` is een standaard-JavaScript-methode die een alertbox op het scherm weergeeft.
- Alle opdrachten binnen de `<script>`-tags moeten eindigen met een puntkomma (;).
- JavaScript kan zowel in de `<head>`- als in de `<body>`-secties worden toegevoegd.

Figuur 3.2
Het resultaat van opgave 1



3.1.2 Foutconsole in JavaScript

Het komt vaak voor dat we tijdens het coderen van onze scripts type- of syntaxfouten maken, waardoor we een leeg scherm te zien krijgen. Dit gebeurt doordat ons script niet goed werkt. Wat we dan moeten doen is teruggaan naar de code van het script om de fouten te verbeteren. Dit noemen we debugging.

Ga terug naar de code van je script interactieve.html en maak een typefout: wijzig **lert** in **lert** en sla het script op.

```
<script type="text/javascript">
    lert("Welcome to my world!!!!");
</script>
```

Alle browsers hebben hulpmiddelen om ons te helpen deze fouten te herstellen. In de volgende figuur zien we de JavaScript-console van de browser Chrome. Om dit scherm te krijgen klik op **Instellingen** daar na op **Extra** en dan kies je **JavaScript-console**. Onder aan de JavaScript-console zien we de volgende foutmelding:

```
Uncaught ReferenceError: lert is not defined
interactieve.html:10
```

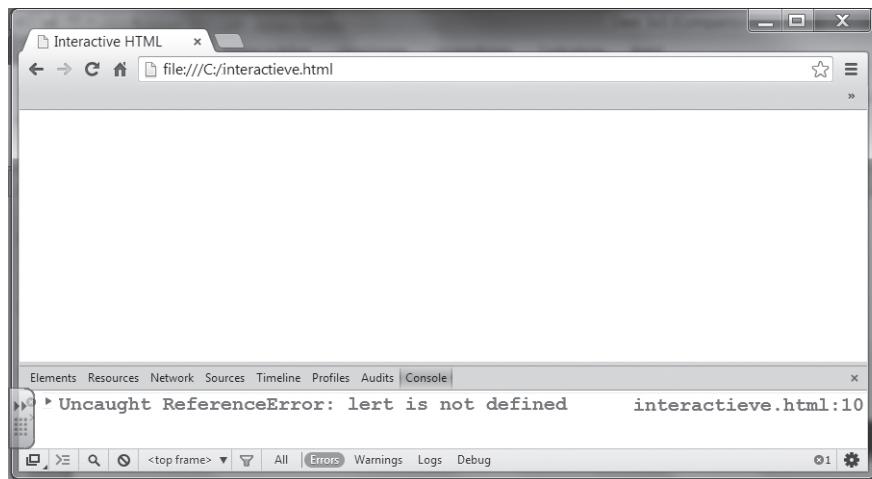
De foutconsole heeft onze typefout gedetecteerd. De typefout is gedetecteerd in regel 10. Dit is vermeld aan de rechterkant:

```
interactieve.html:10
```

Het resultaat is een leeg scherm, zoals te zien is in de volgende figuur. Onderaan zien we in de JavaScript-console of foutconsole de foutmelding staan. **lert** is ongedefinieerd, het is geen ingebouwde methode van JavaScript.

Figuur 3.3

Leeg scherm
met onderaan
de foutconsole



- Opgave** 2 Open **interactieve.html**. Verbeter de syntaxisfout en codeer een alertbox met de volgende tekst: **Opgave 2: Welkom bij mijn eerste JavaScript.**

3.2 De scriptingtaal JavaScript

De JavaScript-syntaxis is hetzelfde als de syntaxis van andere programmeertalen, zoals C en Java. De volgende tabel is een samenvatting van de JavaScript-syntaxis.

Taalcomponent	JavaScript-syntaxis
Scripttags voor openen en sluiten	<script type="text/javascript"> </script>
Blok	{ }
Commentaar	// mijn commentaar /* mijn commentaar */
Declaratie van een variabele	var variabelennaam
Strings	Een string geef je aan tussen aanhalingstekens "Titel JavaScript is nieuw"
Escapeteken \	Aanhalingstekens in een string geef je aan met \' " Titel \"JavaScript\" is nieuw "

3.3 Ingebouwde functies

Alle programmeertalen hebben een ingebouwde *library* (bibliotheek) met *functions* (functies) die de meest voorkomende taken uitvoeren en de werklast van de programmeur verlichten. Zoals de functie `alert()` heeft JavaScript nog meer functies. In de volgende paragrafen noemen we een aantal van deze functies.

3.3.1 De functie document.write()

Om output te kunnen weergeven in een webpagina gebruik je de functie document.write().

Syntaxis:

```
document.write("eigen tekst");
```

Deze functie maakt van je eigen tekst een HTML-tekst. We kunnen ook HTML-tags in onze tekst plaatsen, zoals in opgave 3.

- Opgave** 3 Open **interactieve.html** en voeg in binnen je <script>-tags de volgende code toe.

```
document.write("<p>Opgave 3: Een paragraaf</p>");
```

3.3.2 De functie confirm()

Bij een interactieve webpagina hoort feedback van de gebruiker. Een manier om feedback te vragen is met de functie confirm(). Deze functie genereert een confirmbox waarmee je de user kunt vragen om informatie te verifiëren.

Syntaxis:

```
confirm("jouw vraag?");
```

- Opgave** 4 Open **interactieve.html** en voeg de volgende functie confirm() er aan toe.

```
confirm("Opgave 4: Wilt u doorgaan?");
```

Figuur 3.4
Het resultaat
van opgave 4



- Klik je op OK, dan is het resultaat van de confirmbox `true`.
- Klik je op Cancel of Annuleren, dan is het resultaat van de confirmbox `false`.

3.3.3 De functie prompt()

Met de promptbox vraag je om user input. Als je bijvoorbeeld meer gegevens van de user nodig hebt om een bepaald proces uit te voeren, kun je de user 'prompten' met de vraag voor meer informatie.

Syntaxis:

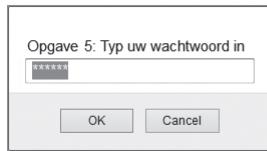
```
prompt("jouw prompttekst", "default waarde");
```

- Opgave** 5 Open **interactive.html** en maak de volgende promptbox:

```
prompt("Opgave 5: Typ uw wachtwoord in","*****");
```

Figuur 3.5

Het resultaat van opgave 5



- Klik je op OK, dan is het resultaat van de promptbox de inputtekst.
- Klik je op Annuleren, dan is het resultaat van de promptbox een nul.

Opgave

- 6 Open **interactive.html** en codeer het volgende:

Stap 1: Codeer de volgende promptbox: **Typ uw gebruikersnaam in**

Stap 2: Codeer de volgende confirmbox: **Wilt u abonnee worden?**

3.4 JavaScript-variabelen

Een JavaScript-variabele is een datacontainer met een naam en een waarde. De naam van de variabele is het adres van de variabele in het computergeheugen. Op dit adres kun je data tijdelijk bewaren. Een variabele kun je voorstellen als een laatje met een naam in het geheugen van de computer. Door middel van de naam kun je de inhoud van dat laatje bekijken of wijzigen.

Een variabele kan de volgende datatypen en waarden hebben:

datatype	voorbeeld
date	11/11/2014
string	“Carl”
boolean	true/false
integer	19
float	3.56

3.4.1 Namen van variabelen

De naam van een variabele mag letters, getallen of underscores hebben. De naam van de variabele moet een beschrijving zijn van de content van de variabele. Bijvoorbeeld:

```
stad = "Amsterdam";
```

De naam is hoofdlettergevoelig (er is onderscheid tussen namen met en zonder hoofdletters). Bijvoorbeeld `Stad` en `stad` zijn twee verschillende variabelen. Hieronder declareren we een variabele met een string:

```
var student = "Carl Jung";
```

Hier mee declareer je de variabele `student` en sla je die op met de content "Carl Jung". Je zou dit als volgt kunnen voorstellen:

student
"Carl Jung"

Hieronder declareren we een variabele met een booleaanse waarde:

```
var leerplichtig = true;
```

Je zou dit als volgt kunnen voorstellen:

```
leerplichtig
```

```
true
```

Hieronder declareren we een variabele met als waarde een getal:

```
var leeftijd = 16;
```

Je zou dit als volgt kunnen voorstellen:

```
leeftijd
```

```
16
```

De waarde van een variabele kun je wijzigen. Bijvoorbeeld:

```
leeftijd = leeftijd + 1;
```

Hier mee verander je de waarde van de variabele `leeftijd` in 17.

```
leeftijd
```

```
17
```

3.4.2 Operatoren van variabelen

Hieronder zie je de belangrijkste rekenkundige operatoren:

operator	betekenis	voorbeeld	Resultaat in prijs
+	optellen	prijs = 10 + 5	15
-	aftrekken	prijs = prijs - 1	14
++	plus 1	prijs++	15
--	min 1	prijs--	14
+=	optellen	prijs += 10 kort voor prijs = prijs + 10	24
-=	aftrekken	prijs -= 10 kort voor prijs = prijs - 10	14
*	vermenigvuldigen	prijs = prijs * 2	28
/	delen	prijs = prijs / 2	14

In de volgende opgave oefenen we rekenen met variabelen.

Opgave 7 Codeer het volgende script en sla het op als **variabelen.html**.

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <meta http-equiv="Content-Type" content="text/html;
        charset=UTF-8">
    <title>Variabelen</title>
</head>
<body>
    <script type="text/javascript">
        var voornaam = "Carl";
        var achternaam = "Petersen";
        var schoolgeld = 1000;
        var boekengeld = 100;
        var bedrag = schoolgeld + boekengeld;
        var studiefinanciering = 600;
        var totaal = bedrag - studiefinanciering;
        var volledigenaam = voornaam + "<br /> " +
            achternaam;
        document.writeln(
            "<p>Opgave 7</p>" + volledigenaam +
            "<br />Totaal: " + totaal);
    </script>
</body>
</html>
```

Het resultaat moet er als volgt uitzien:

Opgave 7

Carl

Peterson

Totaal:500

In het vorige voorbeeld hebben we de operator + op twee manieren gebruikt. Ten eerste hebben we met de operator + het bedrag gerekend als volgt:

```
var bedrag = schoolgeld + boekengeld;
```

Ten tweede hebben we met de operator + drie stukken tekst aan elkaar geplakt:

```
var volledigenaam = voornaam + "<br /> " + achternaam;
```

We zeggen dat de operator + wordt overloaded. Dit betekent dat deze operator opereert in twee contexten: hij kan gebruikt worden voor het optellen van twee variabelen, of hij kan gebruikt worden voor het plakken van twee of meer teksten. De context bepaalt de gebruikswijze van de operator. Bijvoorbeeld:

Geval 1:

```
resultaat = schoolgeld + boekengeld;
// resultaat is het bedrag 500
```

Geval 2: resultaat = voornaam + boekengeld;
 // resultaat is de string "Carl 100"

Bij geval 1 heeft de operator + twee getallen opgeteld.

Bij geval 2 is de context rond de operator + veranderd naar een string voornaam en een getal boekengeld. De operator + heeft een string en een getal aan elkaar geplakt.

Opgave 8 Open **variabelen.html** en maak de volgende variabelen:

- de variabele achternaam met de waarde Bolt;
- de variabele voornaam met de waarde Zakaria;
- de variabele nederlands met de waarde 9;
- de variabele engels met de waarde 8;
- de variabele rekenen met de waarde 7;
- de variabele gemiddeld met het uitgerekende gemiddeld cijfer.

Geef de variabelen weer.

Het resultaat moet er als volgt uitzien:

Opgave 8
Zakaria Bolt
Nederlands: 9
Engels: 8
Rekenen: 7
Gemiddeld: 8

3.5 Datatypes

Zoals elke programmeertaal, heeft JavaScript verschillende datatypes: nummers, teksten, boolean (true of false), null en undefined. Het is belangrijk om goed te begrijpen wanneer en waarom je een bepaald datatype moet gebruiken.

Het datatype: string

Strings zijn tekenreeksen. Een string geef je aan als een tekst tussen aanhalingstekens, bijvoorbeeld “tekst”. De meeste informatie in databases of webapplicaties bestaat in de vorm van teksten. Bijvoorbeeld NAW-gegevens, productinformatie enzovoort.

Opgave 9 Codeer het volgende script en sla het op als **datatypes.html**.

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <meta http-equiv="Content-Type" content="text/html;
        charset=UTF-8">
    <title>Datatypes</title>
</head>
<body>
    <script type="text/javascript">
```

```

var naam = "Carl";
var adres = "Kruislaan 111";
document.write(
"<br />Opgave 9: Datatype van naam is:"+
typeof naam);
</script>
</body>
</html>

```

De opdracht typeof

In bovenstaand voorbeeld hebben we de twee stringvariabelen `naam` en `adres` aangemaakt. Daarna hebben we met behulp van de opdracht `typeof` gevraagd om het datatype van de variabele `naam`.

Het resultaat moet als volgt uitzien.

Opgave 9: Datatype van naam is: string

Het datatype: boolean

Het datatype `boolean` kan maar twee waarden krijgen: `true` of `false` (waar of onwaar). Op basis van de waarde van een `boolean`-variabele kunnen we straks interessante controles programmeren.

Opgave 10 Open **datatypes.html** en voeg de volgende code er aan toe.

```

var leerplichtig = true;
var volwassen = false;
document.write(
"<br />Opgave 10: Datatype van leerplichtig is:"+
typeof leerplichtig);

```

Het resultaat is:

Opgave 10: Datatype van leerplichtig is: boolean

Het datatype: number (integer)

Een integer is een getal zonder decimale punt; het getal 6 is bijvoorbeeld een integer. Een getal met het datatype `number` (`integer`) kan positief of negatief zijn.

Opgave 11 Open **datatypes.html** en voeg de volgende code er aan toe.

```

var temperatuur = -10;
document.write(
"<br />Opgave 11: Datatype van temperatuur is:"+
typeof temperatuur);

```

Het resultaat is:

Opgave 11: Datatype van temperatuur is: number

Het datatype: number (floating-point)

JavaScript behandelt alle getallen als getallen met een ‘floating point’ (drijvende komma). Een floating-pointgetal is dus een getal met een decimale komma; in JavaScript wordt de decimale komma weergegeven met een punt, zoals in het Engels gebruikelijk is. Bijvoorbeeld 19% van 100 codeer je als volgt:

Opgave 12 Open **datatype.html** en voeg de volgende code er aan toe.

```
var btw = 0.19 * 100;  
document.write(  
"<br />Opgave 12: Waarde van btw is: " + btw);
```

Het resultaat 19 is van het datatype floating-point; de decimale punt is naar een andere positie verschoven (‘gedreven’).

Opgave 12: Waarde van btw is: 19

Conversie van tekst naar integer met parseInt()

Met de functie `parseInt()` maak je conversies van tekst-datatypes naar integer-datatypes, bijvoorbeeld:

Opgave 13 Open **datatype.html** en voeg de volgende code er aan toe.

```
var string = "33445";  
var resultaat = parseInt(string);  
document.write(  
"<br />Opgave 13:parseInt(string):::Resultaat is "+  
typeof resultaat);
```

Het resultaat is:

Opgave 13:parseInt(string) :: Resultaat is number

Conversie van tekst naar float met parseFloat()

Met de functie `parseFloat()` maak je conversies van tekst naar float-datatypes, bijvoorbeeld:

Opgave 14 Open **datatype.html** en voeg de volgende codes er aan toe.

```
var string = "3.3445";  
var resultaat = parseFloat(string);  
document.write(  
"<br />Opgave 14: parseFloat(string):::Resultaat is "+  
typeof resultaat);
```

Het resultaat is:

Opgave 14: parseFloat(string) :: Resultaat is number

Conversie van float naar integer met parseInt()

Met de functie `parseInt()` maak je conversies van float-datatypes naar integer-datatypes, bijvoorbeeld:

Opgave 15 Open **datatype.html** en voeg de volgende code er aan toe.

```
var float = 3.3445;
var resultaat = parseInt(float);
document.write(
"<br />Opgave 15: parseInt(float):::Resultaat is "+
typeof resultaat);
```

Het resultaat is:

Opgave 15: parseInt(float) :: Resultaat is number

Conversie van integer of float naar tekst met toString()

Met de functie `toString()` maak je conversies van tekst- naar float-data-types, bijvoorbeeld:

Opgave 16 Open **datatype.html** en voeg de volgende code er aan toe.

```
var float = 3.3445;
var resultaat = float.toString();
document.write(
"<br />Opgave 16: float.toString():::Resultaat is "+
typeof resultaat);
```

Het resultaat is:

Opgave 16: float.toString() :: Resultaat is string

Formatteren naar string met n decimale posities met toFixed(n)

Met de functie `toFixed(n)` formatteer je van float naar *n* decimale posities.

Opgave 17 Open **datatype.html** en voeg de volgende codes er aan toe.

```
var float = 3.3445;
var resultaat = float.toFixed(2);
document.write(
"<br />Opgave 17: float.toFixed(2):::Resultaat is "+
typeof resultaat);
```

Het resultaat is:

Opgave 17: float.toFixed(2) :: Resultaat is string

Het datatype: undefined

Als we een variabele willen gebruiken, moeten we deze variabele eerst definiëren. Een ongedefinieerde variabele krijgt het datatype `undefined`.

Opgave 18 Open **datatypes.html** en voeg de volgende codes er aan toe.

```
var twee;      // variabele twee ongedefinieerd.  
document.write(  
"<br />Opgave 18: Datatype var twee is: " +  
typeof twee);
```

Het resultaat is:

Opgave 18: Datatype var twee is: undefined

Het datatype: NaN

Het datatype `NaN` betekent “Not a Number”. In de volgende opgave (de som van `1 + Null`) krijgt de variabele *drie* het datatype `NaN`:

Opgave 19 Open **datatypes.html** en voeg de volgende code er aan toe.

```
var een = 1;  
var drie = een * twee; // var twee nog niet gedefinieerd.  
document.write(  
"<br />Opgave 19: De waarde van drie is: " + drie);
```

Het resultaat is:

Opgave 19: De waarde van drie is: NaN

3.5.1 Web-lab 01

Codeer het volgende script en sla het op als **web-lab 01.html**.

```
<!DOCTYPE html>  
<html lang="nl">  
<head>  
    <meta http-equiv="Content-Type" content="text/html;  
charset=UTF-8">  
    <title>web-lab 01</title>  
    <!--dit script werkt met variabelen  
auteur:  
datum:  
-->  
</head>  
<body>  
    <h2>Web-lab 01</h2>  
    <h3>Factuur</h3>  
    <script type="text/javascript">  
        var klantnaam = prompt("voer je eigen naam in");  
        var aantal =  
            parseInt(prompt("aantal boeken te bestellen?"));  
        var titel = "JavaScripts";  
        var prijs = 29.90;  
        document.write("<br />Bedankt voor je bestelling: " +  
klantnaam);  
        document.write("<br />Boektitel is: " + titel);  
        document.write("<br />Aantal te bestellen: " + aantal);  
        document.write("<br />Prijs per boek is: " +
```

```
    prijs.toFixed(2) );
</script>
</body>
</html>
```

Het resultaat moet er ongeveer als volgt uitzien:

```
Web-lab 01
Factuur

Bedankt voor je bestelling: Jan
Boektitel is: JavaScripts
Aantal te bestellen: 10
Prijs per boek is: 29.90
```

Lab opgave

Voer nu de volgende opdrachten uit:

1. Maak een nieuwe variabele bedrag.
2. Reken het bedrag uit (bedrag is aantal boeken maal de prijs).
3. Maak een nieuwe variabele btw.
4. Reken de btw van het bedrag uit (btw is 6% van het bedrag).
5. Maak een nieuwe variabele totaal.
6. Reken het totaal te betalen bedrag uit (totaal is bedrag + btw).

Het resultaat moet er als volgt uitzien:

```
Web-lab 01
Factuur

Bedankt voor je bestelling: Peter
Boektitel is: JavaScripts
Aantal te bestellen: 22
Prijs per boek is: 29.90
Bedrag is: 657.80
BTW is: 39.47
Totaal inclusief BTW is: 697.27
```

3.6 Het Array-object

Naast variabelen heb je ook datastructuren nodig waar je informatie tijdelijk in kunt bewaren. Een van de eenvoudigste datastructuren is het Array-object.

3.6.1 Wat is een array?

Een array is een serie van dezelfde soort of van verschillende data-elementen. Ieder element heeft een positie in de array. De positie van een element in de array geef je aan met een index. Een array kun je voorstellen als een ladekastje met een naam in het geheugen van de computer. Elk laatje heeft een eigen volgnummer. Door middel van de naam en het volgnummer (index) kun je de inhoud van een laatje bekijken of wijzigen.

3.6.2 Een Array-object declareren

De syntaxis voor het declareren van een nieuwe array is als volgt:

Syntaxis:

```
var arraynaam = new Array();
```

De `arraynaam` bepalen we zelf. Met het keyword `new` maken we een nieuw object dat gebaseerd is op het JavaScript-object `Array`. Het `Array`-object is een sjabloon of model van een array. Vanuit deze sjabloon kunnen we onze eigen arrays aanmaken.

Opgave 20 Zorg ervoor dat **arrays.html** er als volgt uitziet:

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <meta http-equiv="Content-Type" content="text/html;
        charset=UTF-8">
    <title>Arrays</title>
</head>
<body>
    <script type="text/javascript">
        var weekdag = new Array();
        document.write("<br />Opgave 20: Array aangemaakt");
    </script>
</body>
</html>
```

3.6.3 Array-elementen declareren

Een array mag een of meer elementen hebben. Een element declareer je als volgt:

Syntaxis:

```
weekdag[0] = "zondag";
```

Je zou deze array in het computergeheugen als volgt kunnen voorstellen:

weekdag
[0]
zondag

Een array wordt intern in het computergeheugen aangemaakt, maar niet op het beeldscherm. Als je de content van een array wilt zien, moet je de opdracht `document.write` gebruiken.

Opgave 21 Open **arrays.html** en voeg de volgende code eraan toe:

```
weekdag[0] = "zondag";
weekdag[1] = "maandag";
weekdag[2] = "dinsdag";
weekdag[3] = "woensdag";
weekdag[4] = "donderdag";
```

```
weekdag[5] = "vrijdag";
weekdag[6] = "zaterdag";
document.write("<br />Opgave 21: Elementen aangemaakt");
```

Je zou de array `weekdag` in het computergeheugen als volgt kunnen voorstellen:

weekdag						
[0]	[1]	[2]	[3]	[4]	[5]	[6]
zondag	maandag	dinsdag	woensdag	donderdag	vrijdag	zaterdag

Je verwijst naar een element door de array-naam en de index te gebruiken. Bijvoorbeeld:

```
document.write(weekdag[0]);
```

De indexnummers van een array beginnen altijd met 0. Er is een tweede manier om een array te declareren.

Bovenstaande array zou je ook als volgt kunnen aanmaken:

```
var weekdag = new Array
("zondag", "maandag", "dinsdag", "woensdag", "donderdag",
 "vrijdag", "zaterdag");
```

Opgave 22 Open arrays.html en voeg de volgende code er aan toe:

```
document.write(
"<br />Opgave 22: weekdag array: " + weekdag);
document.write(
"<br />Eerste element in array is: " + weekdag[0]);
```

Het resultaat moet er als volgt uitzien:

```
Opgave 20: Array aangemaakt
Opgave 21: Elementen aangemaakt
Opgave 22: weekdag array: zondag,maandag,dinsdag,woensdag,
donderdag,vrijdag,zaterdag
Eerste element in array is: zondag
```

3.6.4 Multidimensionale arrays

Een array kunnen we als volgt declareren:

```
var team = new Array();
```

Aan deze array kunnen we elementen toevoegen als volgt:

```
team[0] = "Arsenal";
team[1] = "Marseille";
team[2] = "FC Barcelona";
team[3] = "Ajax";
```

Maar een array-element kan zelf ook weer een array zijn, bijvoorbeeld:

```
team[0] = new Array();
```

Om een element van de array team te kunnen verwijzen, hebben we dus twee indexen nodig, bijvoorbeeld:

```
team[0][0] = 3;
```

Hier hebben we een multidimensionale array aangemaakt. Kijk in de volgende tabel naar de scores van drie teams en drie wedstrijden

	Wedstrijd 1	Wedstrijd 2	Wedstrijd 3
Team 1	1	2	3
Team 2	4	5	6
Team 3	7	8	9

Deze gegevens kun je in een array van 3×3 cellen representeren. Eerst declareer je je array team als volgt:

```
var team = new Array();
```

Het resultaat zou je als volgt kunnen voorstellen:

```
team
```

Dan declareer je drie team-elementen. Elk team krijgt een nieuwe array. Bijvoorbeeld:

```
team[1] = new Array();
team[2] = new Array();
team[3] = new Array();
```

Het resultaat zou je als volgt kunnen voorstellen:

```
team
```

```
team[1]
```

```
team[2]
```

```
team[3]
```

Dan voer je de gegevens voor team[1] in. Bijvoorbeeld:

```
team[1][1] = 1;
team[1][2] = 2;
team[1][3] = 3;
```

Het resultaat zou je als volgt kunnen voorstellen:

```
team
```

```
team[1][1]
```

```
team[1][2]
```

```
team[1][3]
```

```
1
```

```
2
```

```
3
```

De eerste index[1] wijst naar het team en de tweede index[1] wijst naar de wedstrijd. Dus bij team 1 wedstrijd 1 was de score 1.

Als je de scores van team 1 wilt zien, doe dat als volgt:

```
document.write("Scores team 1: "+team[1][1]+" "+team[1][2]+"
"+team[1][3]);
Of
document.write("Scores team 1: "+team[1]);
```

3.6.5 Web-lab 02

In de onderstaande tabel zie je de scores van drie teams en drie wedstrijden:

	Wedstrijd 1	Wedstrijd 2	Wedstrijd 3
Team 1	1	2	3
Team 2	4	5	6
Team 3	7	8	9

Maak een bestand **web-lab 02.html** en codeer de volgende drie stappen:

Stap 1: Maak een tweedimensionale array voor de drie teams en drie wedstrijden en met de scores uit bovenstaande tabel.

Stap 2: Bereken de gemiddelde score voor alle drie teams.

Stap 3: Geef de gemiddelde scores weer.

Het resultaat moet er als volgt uitzien:

Web-lab 02

Scores team 1: 1,2,3 Gemiddeld 2

Scores team 2: 4,5,6 Gemiddeld 5

Scores team 3: 7,8,9 Gemiddeld 8

3.7 Array-methodes

Arrays zijn objecten en we gaan nu kennismaken met de array-methodes die ingebouwd zijn in JavaScript. Deze methodes helpen ons om arrays te verwerken en om allerlei bewerkingen op arrays uit te voeren.

Opgave 23 Typ het volgende script over en sla het op als **arraymethodes.html**

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <meta http-equiv="Content-Type"
        content="text/html;
        charset=UTF-8">
    <title>Array methodes</title>
</head>
<body>
    <script>
        var studenten = new Array(
            'Zakaria','Albert','Hamsa','Mike','Pieter');
```

```

document.write('<br />Opgave 23:Studenten:' +
studenten);
</script>
</body>
</html>

```

Het resultaat zie je hieronder.

Opgave 23: Studenten: Zakaria,Albert,Hamsa,Mike,Pieter

Hier hebben we een array-object met vijf elementen aangemaakt ergens in het computergeheugen. De naam van de array is studenten en verwijst naar het geheugenadres waar de array aangemaakt is.

Figuur 3.6

studenten				
[0]	[1]	[2]	[3]	[4]
'Zakaria'	'Albert'	'Hamsa'	'Mike'	'Pieter'

3.7.1 Array-pointers

Array-pointers zijn verwijzingen naar een array-object. Maar pointers zijn geen objecten. In de volgende opgave creëren we de pointer `team` die naar de array `studenten` verwijst. Een pointer is een tweede naam voor een object.

Opgave 24 Open **arraymethodes.html** en voeg de volgende pointer er aan toe.

```

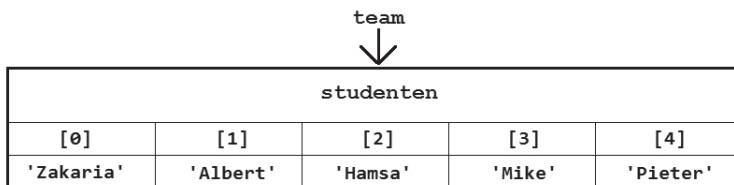
team = studenten;
document.write('<br />Opgave 24: Team:' + team);
</script>
</body>
</html>

```

Het resultaat zie je hieronder. De array `studenten` heeft de tweede naam `team` gekregen.

Opgave 24: Team: Zakaria,Albert,Hamsa,Mike,Pieter

Figuur 3.7



Zoals te zien is in deze figuur, verwijst de pointer `team` naar het array-object `studenten`. Het is geen kopie van het object. Dit gaan we testen met de volgende opgave.

Opgave 25 Open **arraymethodes.html** en voeg de volgende code eraan toe.

```
team[0] = 'Umut';
document.writeln(
'<br />Opgave 25: Element [0] is gewijzigd:'+
studenten );
```

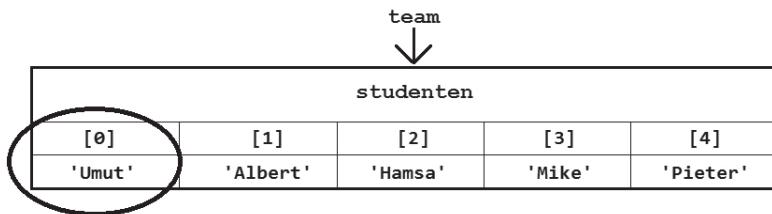
Het resultaat zie je hieronder.

Opgave 25: Element [0] is gewijzigd:Umut,Albert,Hamsa,Mike,Pieter

Je ziet in de volgende figuur dat we nog steeds bezig zijn met hetzelfde object. Met de pointer `team` verwijzen we nog steeds naar het object `studenten`.

```
team[0] = 'Umut';
```

Figuur 3.8



3.7.2 `array.shift()`

Een object mag methodes hebben. Het array-object heeft een aantal handige methodes die ons helpen met het verwerken en manipuleren van arrays. In de volgende opgave gebruiken we de methode `shift()`. Met deze methode kunnen we het eerste element van een array verwijderen.

Opgave 26 Open **arraymethodes.html** en voeg de volgende code er aan toe.

```
studenten.shift();
document.writeln(
'<br />Opgave 26: Resultaat van shift:'+
studenten );
```

Het resultaat zie je hieronder: 'Umut' is verwijderd.

Opgave 26: Resultaat van shift: Albert,Hamsa,Mike,Pieter

Het resultaat van de methode `shift()` is dat de hele array een positie naar links opschuift en het eerste element verdwijnt.

3.7.3 `array.unshift()`

In de volgende opgave gebruiken we de methode `unshift()`. Met deze methode kunnen we een of meer nieuwe element aan het begin van een array toevoegen.

Opgave 27 Open **arraymethodes.html** en voeg de volgende code er aan toe.

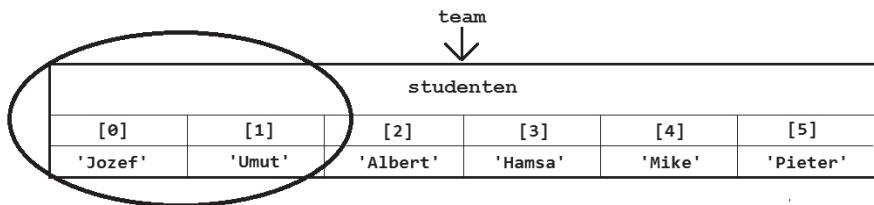
```
studenten.unshift('Jozef','Umut');
document.writeln(
'<br /> Opgave 27: Resultaat van unshift:' +
studenten );
```

Het resultaat zie je hieronder: 'Jozef' en 'Umut' zijn toegevoegd.

Opgave 27: Resultaat van unshift: Jozef,Umut,Albert,Hamsa,Mike,Pieter

Je ziet in de volgende figuur dat er twee nieuwe elementen aan het begin van de array toegevoegd zijn.

Figuur 3.9



3.7.4 array.pop()

In het volgende opgave gebruiken we de methode `pop()`. Met deze methode kunnen we het laatste element uit een array verwijderen.

Opgave 28 Open **arraymethodes.html** en voeg de volgende code er aan toe.

```
studenten.pop();
document.writeln(
'<br /> Opgave 28: Resultaat van pop:' +
studenten );
```

Het resultaat zie je hieronder: 'Peter' is verwijderd.

Opgave 28: Resultaat van pop: Jozef,Umut,Albert,Hamsa,Mike

Het resultaat van de methode `pop()` is dat het laatste element in je array is verwijderd.

3.7.5 array.push()

In de volgende opgave gebruiken we de methode `push()`. Met deze methode kunnen we een of meer nieuwe elementen aan het einde van een array toevoegen.

Opgave 29 Open **arraymethodes.html** en voeg de volgende code er aan toe.

```
studenten.push('Pieter','David');
document.writeln(
'<br />Opgave 29:Resultaat van push: ' +
studenten );
```

Het resultaat zie je hieronder: 'Peter' en 'David' zijn toegevoegd.

Opgave 29: Resultaat van push: Jozef,Umut,Albert,Hamsa,Mike,Pieter,David

Je ziet in de volgende figuur dat er twee nieuwe elementen aan het einde van de array toegevoegd zijn.

Figuur 3.10

studenten						
[0]	[1]	[2]	[3]	[4]	[5]	[6]
'Jozef'	'Umut'	'Albert'	'Hamsa'	'Mike'	'Pieter'	'David'

3.7.6 array.splice()

In de volgende opgave gebruiken we de methode `splice()`. Met deze methode kunnen we een array in tweeën splitsen.

Syntaxis van de methode `splice()`:

```
arraynaam.splice(begin-element, aantal-elementen)
```

Als je bijvoorbeeld een nieuwe array wilt maken met de eerste drie elementen uit de array `team`, dan doe je dat zo:

```
nieuwearray = team.splice(0,3)
```

Opgave 30 Open **arraymethodes.html** en voeg de volgende code er aan toe.

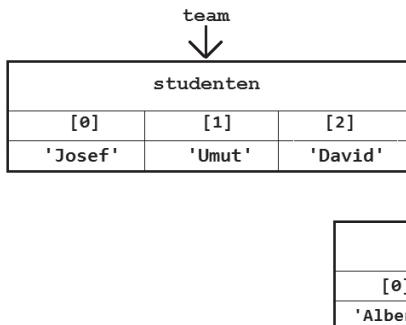
```
klas = studenten.splice(2,4);
document.writeln(
'<br />Opgave 30:Resultaat van splice ----studenten:'+
studenten+'---klas:'+klas);
```

Het resultaat zie je hieronder.

Opgave 30: Resultaat van splice ---- studenten: Jozef,Umut,David---- klas:
Albert,Hamsa,Mike,Pieter

Je ziet in de volgende figuur dat de array `studenten` gesplitst is in twee arrays. Eerst krijgt de array `klas` vier elementen, beginnend met element[2]. Dan krijg je in de array `studenten` de overgebleven drie elementen.

Figuur 3.11



3.7.7 array.sort()

In de volgende opgave gebruiken we de methode `sort()`. Met deze methode kunnen we een array ordenen in alfabetische volgorde.

Opgave 31 Open **arraymethodes.html** en voeg de volgende code er aan toe.

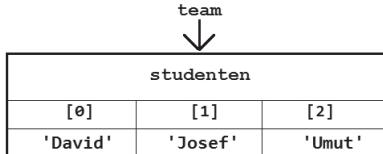
```
studenten.sort();
document.writeln(
  '<br />Opgave 31:Resultaat van sort:' +
  studenten );
```

Het resultaat zie je hieronder.

Opgave 31: Resultaat van sort: David,Jozef,Umut

Je ziet dat de volgorde van de elementen is veranderd.

Figuur 3.12



3.7.8 array.reverse()

In de volgende opgave gebruiken we de methode `reverse()`. Met deze methode kunnen we een array ordenen in de omgekeerde volgorde.

Opgave 32 Open **arraymethodes.html** en voeg de volgende code er aan toe.

```
studenten.reverse();
document.writeln(
  '<br />Opgave 32:Resultaat van reverse:' +
  studenten);
```

Het resultaat zie je hieronder.

Opgave 32: Resultaat van reverse: Umut,Jozef,David

3.7.9 array.indexOf()

Met de methode `indexOf()` kunnen we kijken of er een bepaald element in een array bestaat. We zouden bijvoorbeeld kunnen vragen of een bepaalde student in de array studenten bestaat.

Syntaxis van de methode `indexOf()`:

```
var index = arraynaam.indexOf('element')
```

Als het element bestaat, krijg je de juiste index van het element. Als het element niet bestaat, krijg je (-1) als resultaat.

Opgave 33 Open **arraymethodes.html** en voeg de volgende code er aan toe.

```
index = studenten.indexOf('Jozef');
document.writeln(
'<br />Opgave 33:Resultaat van indexOf: ' +
index );
```

Het resultaat zie je hieronder.

Opgave 33: Resultaat van indexOf: 1

3.7.10 Een element verwijderen

De opdracht delete

Om een specifiek array-element te verwijderen, gebruiken we de opdracht `delete`. In de volgende opgave gebruiken we de opdracht `delete` om het derde element in de array `klas` te verwijderen:

Opgave 34 Open **arraymethodes.html** en voeg de volgende code er aan toe.

```
delete klas[2];
document.writeln(
'<br />Opgave 34:Resultaat van delete:' + klas);
```

Het resultaat zie je hieronder.

Opgave 34: Resultaat van delete:Albert,Hamsa,,Pieter
--

Je ziet dat de opdracht `delete` alleen maar de content '`Mike`' van het derde element heeft verwijderd, het derde element blijft staan, maar is leeg tussen '`Hamsa`', '`Pieter`'

Figuur 3.13

klas			
[0]	[1]	[2]	[3]
'Albert'	'Hamsa'		'Pieter'

Dat kunnen we overigens controleren met de opdracht `typeof`.

De opdracht typeof

Met de opdracht `typeof` kunnen we checken wat voor datatype een element of een variabele is.

Opgave 35 Open **arraymethodes.html** en voeg de volgende code er aan toe.

```
document.writeln(
'<br />Opgave 35: Datatype van gedelete element is:'+
typeof klas[2]);
```

Het resultaat zie je hieronder.

Opgave 35: Datatype van gedelete element is:undefined

Hier kunnen we zien dat het datatype van de gedelete element ‘undefined’ is.

3.7.11 Elementen verwijderen met splice()

Om een specifiek array-element volledig te verwijderen, gebruiken we de methode `splice()`.

Syntaxis van de methode `splice()`:

```
arraynaam.splice(element-index, 1)
```

Opgave 36 Open **arraymethodes.html** en voeg de volgende code er aan toe.

```
klas.splice(2,1);
document.writeln(
'<br />Opgave 36: Element volledig verwijderd:'+
klas );
```

Het resultaat zie je hieronder.

Opgave 36: Element volledig verwijderd: Albert,Hamsa,Pieter

Zoals te zien is in de volgende figuur is het derde element volledig verwijderd.

Figuur 3.14

klas		
[0]	[1]	[2]
'Albert'	'Hamsa'	'Pieter'

In de volgende opgave gebruiken we de methodes `splice()` en `indexOf()` om een specifieke student te verwijderen.

Opgave 37 Open **arraymethodes.html** en voeg de volgende code er aan toe.

```
index = studenten.indexOf('Jozef');
studenten.splice(index,1);
document.writeln(
'<br />Opgave 37: Specifieke student verwijderd:'+
studenten );
```

Hier zie je het resultaat: ‘Jozef’ is verwijderd.

Opgave 37: Specifieke student was verwijderd: Umut,David

3.7.12 Een array kopiëren met slice()

Soms wil je een exacte kopie van een array maken. Dat kan met behulp van de methode `slice()` zonder indexen.

Opgave 38 Open `arraymethodes.html` en voeg de volgende code er aan toe.

```
klas2 = klas.slice();
document.writeln(
    '<br />Opgave 38: Resultaat van kopiëren:' +
    klas2 );
```

Als je de methode `slice()` zonder begin-index gebruikt, wordt de eerste index gebruikt. Als je de methode gebruikt zonder eind-index wordt de laatste index gebruikt.

Het resultaat zie je hieronder:

Opgave 38: Resultaat van kopiëren:Albert,Hamsa,Pieter

3.7.13 Een array transporteren

Soms wil je een array naar de server transporteren. Om dit te realiseren converteren we de array in een string. Daarna kunnen we de string in de vorm van een cookie naar de server transporteren. We behandelen cookies aan het einde van dit hoofdstuk.

3.7.14 array.join(delimeter)

De methode `array.join(delimeter)` converteert een array in een string. In de string worden de elementen met het aangegeven delimeter-teken van elkaar gescheiden.

In de volgende opgave maken we gebruik van de methode `array.join(delimeter)` om de array `klas` in een string te converteren.

Opgave 39 Open `arraymethodes.html` en voeg de volgende code er aan toe.

```
var myString =  klas.join('|');
document.writeln(
    '<br />Opgave 39: Resultaat van join:' +
    myString );
```

Het resultaat is de volgende string:

Opgave 39: Resultaat van join is: ‘Albert|Hamsa|Peter’

3.7.15 Web-lab 03

Maak een nieuw script (**web-lab 03.html**) en voer de volgende stappen uit.

1. Codeer de volgende array met de naam ‘testarray’.

testarray			
[0]	[1]	[2]	[3]
‘een’	‘twee’	3	‘vier’

2. Voeg het vijfde element met de waarde 5 er aan toe.
3. Toon (testarray).
4. Toon het datatype van het vijfde element.
5. Verwijder het vijfde element.
6. Toon (testarray).
7. Voeg in het begin een element in met de waarde ‘nul’.
8. Toon (testarray).
9. Delete index[0].
10. Toon het datatype van index[0].
11. Verwijder het eerste element.
12. Toon (testarray).
13. Verwijder het tweede element.
14. Toon (testarray).
15. Als index[2] bestaat: verwijder het.
16. Toon (testarray).

Als alles goed verloopt, zal het resultaat er als volgt uitzien:

Web-lab 03

```
Stap 3. een,twee,3,vier,5
Stap 4. Datatype is: number
Stap 6. een,twee,3,vier
Stap 8. nul,een,twee,3,vier
Stap 10. Datatype [0] is: undefined
Stap 12. een,twee,3,vier
Stap 14. een,3,vier
Stap 16. een,3
```

3.8 Hash-arrays

Een hash-array is een array met elementen die in plaats van een index een naam krijgen. Bijvoorbeeld, in plaats van:

```
arraynaam[1]
```

kunnen we ook zeggen:

```
arraynaam['elementnaam']
```

JavaScript kent geen hash-arrays, maar we kunnen een hash-array als volgt simuleren:

- Opgave 40** Open een nieuw script en sla het op als **hasharrays.html**. Voeg de volgende code er aan toe.

```
<!DOCTYPE html>
<html lang="nl">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
            charset=UTF-8">
        <title>Hash-arrays</title>
    </head>
    <body>
        <script>
            var laptops = new Array();
            laptops['macbook'] =
                {'model':'air', 'voorraad':2000,'prijs':1090};
            document.write(
                '<br />Opgave 40: laptops array aangemaakt');
        </script>
    </body>
</html>
```

Het resultaat is:

Opgave 40: laptops array aangemaakt

In bovenstaand script hebben we een array `laptops` aangemaakt. Daarna hebben we als volgt een hash-array `laptops` gesimuleerd:

```
laptops['macbook'] =
    {'model':'air', 'voorraad':2000,'prijs':1090};
```

In dit geval is `['macbook']` geen element van de array `laptops` maar een property van de array `laptops`. De array `laptops` is nog steeds leeg. We hebben er alleen de nieuwe property `['macbook']` toegevoegd. Deze property hebben we verder als een nieuw object gedefinieerd. Een nieuw object coderen we ook met twee accolades:

```
{ }
```

Voor dit object hebben we weer drie nieuwe properties met waarden gedefinieerd:

```
{'model':'air', 'voorraad':2000, 'prijs':1090}
```

Omdat onze array `laptops` leeg is, kunnen we de array-methodes zoals `pop()` en `push()` uit de vorige paragraaf niet hier gebruiken. Maar we kunnen dit verder beschouwen als een hash-array.

Opgave 41 Open **hasharrays.html** en voeg de volgende laptops er aan toe.

```
laptops['asusbook'] =  
{'model':'wind', 'voorraad':1000, 'prijs':990};  
laptops['dellbook'] =  
{'model':'fire', 'voorraad':987, 'prijs':890};  
document.write(  
'<br />Opgave 41: laptops array uitgebreid');  
</script>  
</body>  
</html>
```

Het resultaat is:

Opgave 41: laptops array uitgebreid

3.8.1 array.length

We kunnen verder testen of inderdaad onze array `laptops` leeg is met behulp van de property `array.length`.

Opgave 42 Open **hasharrays.html** en voeg de volgende code er aan toe.

```
document.write(  
'<br />Opgave 42: De laptops array heeft '+  
laptops.length + ' elementen');
```

Het resultaat is:

Opgave 42: De laptops array heeft 0 elementen

3.8.2 Testen op array-properties

We kunnen nu gaan testen of een array een bepaalde property of object heeft. Dit doen we met de volgende syntaxis:

propertynaam in arraynaam

Bijvoorbeeld:

```
var mac = 'macbook' in laptops;
```

Als de property in de array bestaat, dan is het resultaat `true`, anders `false`.

Opgave 43 Open **hasharrays.html** en voeg de volgende code er aan toe.

```
var mac = 'macbook' in laptops;  
var asus = 'asusbook' in laptops;  
var dell = 'dellbook' in laptops;  
document.write(  
'<br />Opgave 43: De variabele mac is: ' + mac);  
  
document.write(  
'<br />Mac specificaties ----- '+  
'<br />Model: ' + laptops['macbook']['model'] +  
'<br />Voorraad: ' + laptops['macbook']['voorraad'] +  
'<br />Prijs: ' + laptops['macbook']['prijs']);
```

Het resultaat moet er als volgt uitzien:

Opgave 43: De variabele mac is: true

Mac specificaties -----

Model: air

Voorraad: 2000

Prijs: 1090

3.8.3 Web-Lab 04

1. Maak een nieuw JavaScript en sla het op als **web-lab 04.html**
2. Codeer een nieuwe array met de naam `palet` en met vijf kleuren.
3. Toon (`palet`).
4. Codeer de volgende prompt:
`var nieuwekleur = prompt("Typ een nieuwe kleur in:");`
5. Codeer een array-methode die de nieuwe kleur aan het einde van de array `palet` toevoegt.
6. Toon (`palet`).
7. Codeer de volgende prompt:
`var nieuwekleur = prompt("Typ een nieuwe kleur in:");`
8. Codeer een array-methode die de nieuwe kleur aan het begin van array `palet` toevoegt.
9. Toon (`palet`).
10. Sorteer de array `palet` op oplopend alfabetische volgorde.
11. Toon (`palet`).
12. Splits de array `palet` in tweeën: `palet` en `palet2`.
13. Toon (`palet`).
14. Toon (`palet2`).
15. Maak een exacte kopie van `palet` in `palet3`.
16. Toon (`palet3`);

Hier onder zie je het resultaat met willekeurige kleuren:

Web-lab 04

Stap 3. `palet` is nu: wit,blauw,groen,rood,zwart

Stap 6. `palet` is nu: wit,blauw,groen,rood,zwart,geel

Stap 9. `palet` is nu: oranje,wit,blauw,groen,rood,zwart,geel

Stap 11. `palet` is nu: blauw,geel,groen,oranje,rood,wit,zwart

Stap 13. `palet` is nu: rood,wit,zwart

Stap 14. `palet2` is nu: blauw,geel,groen,oranje

Stap 16. `palet3` is nu: rood,wit,zwart

3.9 Het Date-object

Het Date-object is ook een interessant object in JavaScript. Dit object heeft methodes die ons helpen om een datum te verwerken.

3.9.1 Een Date-object declareren

De syntaxis voor het declareren van een Date-object is als volgt:

Syntaxis

```
new Date();
```

Met het keyword `new` maken we een nieuw object dat gebaseerd is op het JavaScript Date-object. Dit nieuwe object bevat de datum en de tijd van je computerklok.

- Opgave 44** Open een nieuw bestand en sla het op als **dateobject.html**. Voeg de volgende code er aan toe.

```
<!DOCTYPE html>
<html lang="nl">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
            charset=UTF-8">
        <title>Date Object</title>
    </head>
    <body>
        <script type = "text/javascript">
            var vandaag = new Date();
            document.write('<br />Opgave 44: ' + vandaag);
        </script>
    </body>
</html>
```

De huidige datum en tijd worden weergegeven. Bijvoorbeeld:

Opgave 44: Wed Nov 13 2013 14:56:56 GMT+0100

In het resultaat zien we de datum en de tijd van vandaag. De tijd wordt gelezen vanuit je computerklok en is lokale tijd. Lokale tijd in Nederland is CET afkorting voor Central European Time. CET is een uur later dan GMT, dus (GMT+0100). GMT is afkorting voor Greenwich Mean Time. De Royal Observatory in Greenwich is waar alle tijdzones worden gemeten.

Een datum is een moment in de tijd. Dit moment wordt gerekend als het aantal milliseconden sinds middernacht 1 januari 1970. Bijvoorbeeld,

```
var nu = Date.now();
```

geeft als resultaat het aantal milliseconden sinds 1 januari 1970.

Een dag bevat 86.400.000 milliseconden, dus we kunnen zeggen:

```
var eenDag = 86400000;
```

3.9.2 Creëer een Date-object met parameters

Een nieuw Date-object kunnen we creëren met meerdere parameters. In de volgende opgave creëren we een Date-object met de parameter `value`.

Syntaxis met de parameter `value`:

```
new Date(value);
```

De datum van morgen kunnen we als volgt uitrekenen:

```
var morgen = new Date(nu + eenDag);
```

Opgave 45 Open **dateobject.html** en voeg de volgende code er aan toe.

```
var nu = Date.now();
var eenDag = 86400000;
var morgen = new Date(nu + eenDag);
document.write(
    "<br />Opgave 45: De datum morgen is: " + morgen);
</script>
</body>
</html>
```

Afhankelijk van de datum vandaag ziet het resultaat eruit als:

Opgave 45: De datum morgen is: Thu Nov 14 2013 15:01:59 GMT+0100

3.9.3 Creëer een Date-object met string

Weer een andere manier om een Date-object te creëren is met een string-parameter. De syntaxis met de parameter `dateString` is als volgt:

```
new Date(dateString);
```

In de volgende opgave maken we een nieuw Date-object met een `dateString`.

Opgave 46 Open **dateobject.html** en voeg de volgende code er aan toe.

```
var schooljaar = new Date("2014-09-04");
// met tijd
var wintertijd = new Date("2014-10-28T03:00:00");
document.write(
    "<br />Opgave 46: Begin wintertijd is: " +
    wintertijd);
// maak hieronder een var met eigen geboortedatum:
// geef hieronder je geboortedatum weer:
```

Het resultaat moet er ongeveer uitzien als:

Opgave 46: Begin wintertijd is: Tue Oct 28 2014 03:00:00 GMT+0100
--

Mijn geboortedatum is: Wed Oct 01 1980 02:00:00 GMT+0200
--

3.9.4 Creëer een Date-object met getallen

Weer een andere manier om een Date-object te creëren is met getallen. De syntaxis met getallenparameters is als volgt:

```
new Date(jaar,maand[,dag,uur,minuut,seconde,milliseconde])
```

Opgave 47 Open **dateobject.html** en voeg de volgende code er aan toe.

```
beginSchooldag = new Date(2013,8,9,8,30);
document.write(
"<br />Opgave 47: Begin schooldag: " +
beginSchooldag);
```

Het resultaat moet er ongeveer zo uitzien:

```
Opgave 47: Begin schooldag: Mon Sep 09 2013 08:30:00 GMT+0200
```

Opgave 48 Open **dateobject.html** en voeg de volgende code er aan toe.

```
nu = Date.now();
document.write("<br />Opgave 48: Nu: " + nu);
var dagen = ((nu -beginSchooldag) / eenDag);
document.write(
"<br />Aantal dagen sinds begin schooldag : " +
dagen.toFixed(0));
```

Afhankelijk van de datum vandaag moet het resultaat er uitzien als:

```
Opgave 48: Nu: 1384353384807
Aantal dagen sinds begin schooldag : 65
```

3.9.5 Web-lab 05

Maak een nieuw script (**web-lab 05.html**) en voer de volgende stappen uit.

1. Codeer de volgende prompt:

```
var einddatum = prompt('Wat is de einddatum JJJJ-MM-DD?');
```

2. Codeer een Date-object var deadline met de einddatum.

3. Toon de datum in deadline.

4. Codeer een var nu met Date.now().

5. Codeer de var dag = 86400000.

6. In deze stap ga je de **resterende** dagen tussen nu en de deadline uitrekenen.

7. Toon het volgende:

```
document.write("<br />Nog : " + dagen.toFixed(0) + ' dagen
te gaan!');
```

Afhankelijk van de datum vandaag moet het resultaat er zo uitzien:

```
Web-lab 05
Deadline:Sat Dec 12 2020 01:00:00 GMT+0100
Nog : 2584 dagen te gaan!
```

3.9.6 Date-methodes

Het Date-object bevat een aantal methodes die ons helpen om datums te verwerken. Hier volgt een lijst met de meest gebruikelijk Date-methodes:

Methode	Betekenis
getFullYear	geeft het jaar als resultaat (JJJJ)
getMonth	resultaat is de maand (0-11) januari is 0
getDate	resultaat is de dag van de maand (1-31)
getDay	resultaat is de dag van de week (0-6). zondag is 0
getHours	resultaat is het uur (0-23)
getMinutes	resultaat is minuten (0-59)
getSeconds	resultaat is seonden (0-59)
getTime	resultaat is milliseconden sinds 1 januari 1970
now	resultaat is milliseconden sinds 1 januari 1970
toDateString	resultaat is een leesbare datum
toTimeString	resultaat is een leesbare tijd
setFullYear	jaar resetten (JJJJ)
setMonth	maand resetten (0-11)
setDate	dag resetten (1-31)
setHours	uur resetten (0-24)
setMinutes	minuten resetten (0-60)
setSeconds	seonden resetten (0-60)

Met behulp van deze methodes kunnen we met een Date-object werken. Bijvoorbeeld:

```
var vandaag = new Date();
var jaar = vandaag.getFullYear();
vandaag.setHours(0);
```

3.9.7 Web-lab 06

Maak een nieuw script (**web-lab 06.html**) en voer de volgende stappen uit.

1. Codeer de volgende twee arrays:

```
var maanden = new Array(
    'januari','februari','maart','april','mei','juni','juli',
    'augustus','september','oktober','november','december');
var weekdag = new Array(
    'zondag','maandag','dinsdag','woensdag','donderdag',
    'vrijdag','zaterdag')
```

2. Maak gebruik van deze twee arrays om de systeemdatum in het Nederlands te vertalen.

Afhankelijk van de datum vandaag, moet het resultaat er zo uitzien:

Vandaag is: Mon Oct 28 2013 14:41:20 GMT+0100

Vandaag is: maandag 28 oktober 2013

3.9.8 Web-Lab 07

1. Open **web-lab 01.html** en sla het op als **web-lab 07.html**.
2. Maak een nieuw Date-object.
3. Geef de datum van vandaag weer als factuurdatum:dd-mm-jjjj.
4. Reken betaaldatum uit als factuurdatum + 5 dagen.
5. Geef de betaaldatum weer als dd-mm-jjjj.

Het resultaat moet er als volgt uitzien:

Web-lab 07

Factuur

Factuurdatum: dd-mm-jjjj

Betaaldatum: dd-mm-jjjj

Bedankt voor je bestelling: Peter

Boektitel is: JavaScripts

Aantal te bestellen: 22

Prijs per boek is: 29.90

Bedrag is: 657.80

BTW is: 39.47

Totaal inclusief BTW is: 697.27

3.10 Beslissingsstructuren

Alle programmeertalen hebben beslissingsstructuren. Wanneer we een beslissingsstructuur gebruiken, bepalen we de volgorde van uitvoering van de instructies in een programma. Om beslissingsmomenten duidelijk aan te geven gebruiken we stroomdiagrammen.

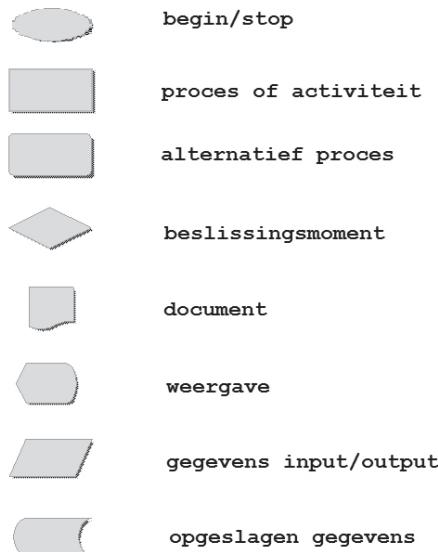
3.10.1 Stroomdiagrammen

Er zijn verschillende soort diagrammen die de communicatie tussen programmeurs en andere specialisten zoals informatieanalisten vereenvoudigen. Het eenvoudigste diagram is het stroomdiagram. De basiselementen van een stroomdiagram zie je in afbeelding 3.15.

3.10.2 De opdracht if

Een beslissingsmoment kunnen we aangeven met de opdracht `if`. Je gebruikt de opdracht `if` om te beslissen wat er moet gebeuren als een voorwaarde `true` of `false` is.

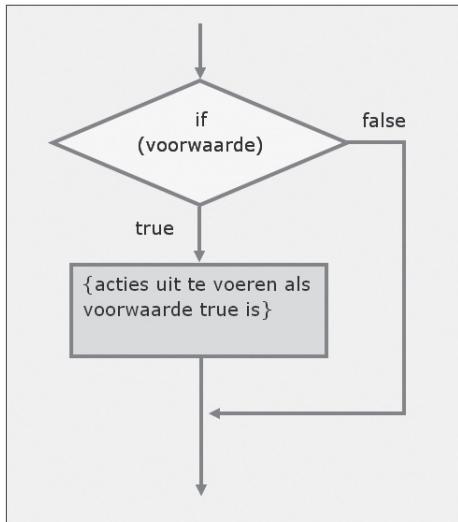
Figuur 3.15
Basiselementen
van het
stroombewerking



De syntaxis van de opdracht **if** is als volgt:

```
if (voorwaarde)
{
    acties uit te voeren als voorwaarde true is;
}
```

Figuur 3.16
Stroombewerking
van de opdracht if



Een voorwaarde in de opdracht **if()** kunnen we aangeven door twee waarden te vergelijken. Dit doen we met vergelijkingsoperatoren. Bijvoorbeeld:

```
if(a == b)
```

De gebruikelijke vergelijkingsoperatoren zie je hieronder:

Symbol	Betekenis	Voorbeeld
==	Gelijk aan	if(leeftijd == 16)
====	Identiek aan	if(naam==== "Carl")
!=	Niet gelijk aan	if(naam != "Jan")
>	Groter dan	if(leeftijd > 17)
>=	Groter dan of gelijk aan	if(leeftijd >= 16)
<=	Kleiner dan of gelijk aan	if(leeftijd <= 16)
&&	AND	if(naam == "Carl" && leeftijd > 16)
	OR	if(naam == "Carl" leeftijd == 16)

- Opgave 49** Open een nieuw bestand en sla het op als **if-opdrachten.html**. Voeg de volgende code er aan toe.

```
<!DOCTYPE html>
<html lang="nl">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
        charset=UTF-8">
        <title>if-opdrachten</title>
    </head>
    <body>
        <script>
            doorgaan = confirm("Opgave 49: Doorgaan?");
            if(doorgaan == true)
            {
                alert("Welkom bij de if-opdracht");
            }
        </script>
    </body>
</html>
```

3.10.3 De clausule else

De clausule `else` is optioneel. De clausule `else` is een alternatieve optie van de opdracht `if`.

De syntaxis van de clausule `else` is als volgt:

```
if (voorwaarde)
{
    acties uit te voeren als voorwaarde true is;
}
else
{
    acties uit te voeren als voorwaarde false is;
}
```

Opgave 50 Open **if-opdrachten.html** en voeg de volgende code er aan toe.

```
if(doorgaan == true)
{
    alert("Opgave 50: Klik op OK om te beginnen");
}
else
{
    alert("Tot ziens!");
}

</script>
</body>
</html>
```

3.10.4 AND-vergelijking

Bij de AND-vergelijking is het hele `if()`-argument alleen `true` als beide argument1 AND argument2 `true` zijn.

Bijvoorbeeld:

```
if (uur==24 && minuut>=0)
{
    alert("het is middernacht");
}
```

Hier krijg je de alertbox te zien alleen als `uur` gelijk is aan `24` en als ook `minuut` gelijk of groter is aan `0`. Beide voorwaarden moeten waar zijn.

Beslissingstabel van AND

Argument 1	AND	Argument 2	resultaat
(true	&&	true)	true
(true	&&	false)	false
(false	&&	false)	false
(false	&&	true)	false

3.10.5 OR-vergelijking

Bij een OR-vergelijking is het hele `if()`-argument `true` als één of beide argumenten `true` zijn.

Bijvoorbeeld:

```
var cola = true;
var thee = false;
if (cola || thee)
{
    alert("we mogen of cola of thee of allebei");
```

Beslissingstabel van OR

Argument 1	OR	Argument 2	resultaat
(true		true)	true
(true		false)	true
(false		false)	false
(false		true)	true

Opgave 51 Open **if-opdrachten.html** en voeg de volgende code er aan toe.

```

if(doorgaan == true)
{
    leeftijd = parseInt(prompt(
        "Opgave 51: Wat is je leeftijd?"));
    if(leeftijd < 17)
    {
        leerplichtig = true;
    }
    else
    {
        leerplichtig = false;
    }
    gediplomeerd = confirm('Heb je je diploma behaald?');

    if(leerplichtig && !gediplomeerd)
    {
        alert(
            'je bent leerplichtig EN zonder diploma: Je moet ' +
            'door studeren');
    }
    if(!leerplichtig && !gediplomeerd)
    {
        alert(
            'je bent niet meer leerplichtig ' +
            'MAAR zonder diploma. ' +
            'Advies: door studeren');
    }
    if(!leerplichtig || gediplomeerd)
    {
        alert(
            'je bent niet meer leerplichtig ' +
            'OF je hebt een diploma: ' +
            'OF allebei');
    }
}
}
</body>
</html>
```

In bovenstaand script voeren we met de if-opdracht drie verschillende testen uit. Ten eerste testen we of de user leerplichtig is EN geen diploma heeft:

```
if(leerplichtig && !gediplomeerd)
```

Ten tweede testen we of de user niet leerplichtig is EN geen diploma heeft:

```
if(!leerplichtig && !gediplomeerd)
```

Als laatste testen we of de user niet leerplichtig is OF een diploma heeft:

```
if(!leerplichtig || gediplomeerd)
```

Dit zijn drie verschillende if-opdrachten. Maar we kunnen in feite alle testen binnen één if-opdracht uitvoeren met de clausule else-if.

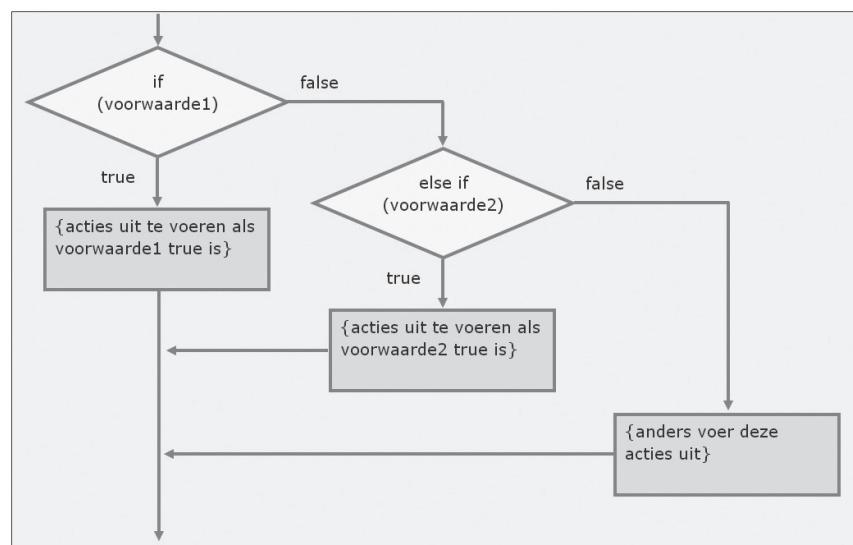
3.10.6 De clausule else-if

De clausule else-if is ook optioneel en geeft ons de mogelijkheid meerdere alternatieve beslissingen te testen. De syntaxis van de clausule else-if is als volgt:

```
if (voorwaarde1)
{
    acties uit te voeren als voorwaarde1 true is;
}
else if (voorwaarde2)
{
    acties uit te voeren als voorwaarde2 true is;
}
else
{
    anders voer deze acties uit;
}
```

Figuur 3.17

Stroomdiagram van de clausule else-if



Opgave 52 Open **if-opdrachten.html** en voeg de volgende else-if-clausules er aan toe.

```
if(leerplichtig && !gediplomeerd)
{
    document.write(
        'leerplichtig EN OOK zonder diploma:doorstudereren');
```

```
        }
    else if(!leerplichtig && !gediplomeerd)
    {
        document.write(
            'je bent niet meer leerplichtig ' +
            'MAAR zonder diploma. ' +
            'Advies: door studeren');
    }
else if(!leerplichtig || gediplomeerd)
{
    document.write(
        'je bent niet meer leerplichtig ' +
        'OF je hebt een diploma: ' +
        'OF allebei');
}
```

3.10.7 Web-Lab 08

1. Open **web-lab 07.html** en sla het op als **web-lab 08.html**.
2. Maak een nieuwe variabele met de naam `bezorgkosten = 15;`
3. Maak een nieuwe variabele met de naam `woonplaats`.
4. Vraag ('prompt') de gebruiker om zijn woonplaats.
5. Maak een nieuwe variabele met de naam `postcode`.
6. Vraag ('prompt') de gebruiker om zijn postcode.
7. Als de woonplaats **Amsterdam** is en de postcode ligt tussen **1000AA** en **2000BB**, dan moet de user de volgende alertbox krijgen: **U betaalt geen bezorgkosten**. De variabele **bezorgkosten** moet **nul** zijn.
8. Totaal te betalen moet totaal te betalen plus **bezorgkosten** zijn.

Als de woonplaats Amsterdam is en de postcode ligt tussen 1000AA en 2000BB, dan betaalt de klant dus geen bezorgkosten.

Een mogelijk resultaat zie je hieronder:

Web-lab 08
Factuur
Factuurdatum: dd-mm-jjjj
Betaaldatum: dd-mm-jjjj
Bedankt voor je bestelling: Jozef
WoonplaatsAmsterdam
Postcode1090ac
BoektitelJavaScripts
Aantal te bestellen.... 10
Prijs per boek.....29.90
Bedrag299.00
BTW..... 17.94
Bezorgkosten0
Totaal inclusief BTW en bezorgkosten316.94

Tip:

Gebruik de functies `toLowerCase()` en `toUpperCase()` om een string om te zetten in hoofdletters of in kleine letters, bijvoorbeeld:

```
< script type="text/javascript">
    var stad = "Amsterdam";
    stad = stad.toLowerCase(); //resultaat is "amsterdam"
    stad = stad.toUpperCase(); //resultaat is "AMSTERDAM"
</script>
```

3.10.8 De ternary-operator (?:)

De ternary-operator is een verkorting van de if-opdracht. De ternary-operator gebruik je bij if-opdrachten die kort en simpel te schrijven zijn. De syntaxis is als volgt:

(voorwaarde ? resultaat als true : resultaat als false)

Na de voorwaarde komt een vraagteken (?). Daarna komt het resultaat als voorwaarde waar (true) is. Dan volgt een dubbele punt (:). Daarna komt het resultaat als voorwaarde niet waar (false) is. Bijvoorbeeld:

```
< script type="text/javascript">
    var leeftijd = 18;
    var leerplichtig = (leeftijd < 17 ? true : false);
</script>
```

Het resultaat van deze test is: `leerplichtig = false`

In de volgende opgave is het eerste resultaat doorgegeven aan de variabele waardering:

```
< script type="text/javascript">
    var cijfer = 8;
    var waardering=(cijfer>=6?"voldoende":"onvoldoende");
</script>
```

3.10.9 Web-lab 09

Open web-lab **08.html** en sla het op als **web-lab 09.html**.

Codeer een ternary-operator met de volgende functionaliteit:

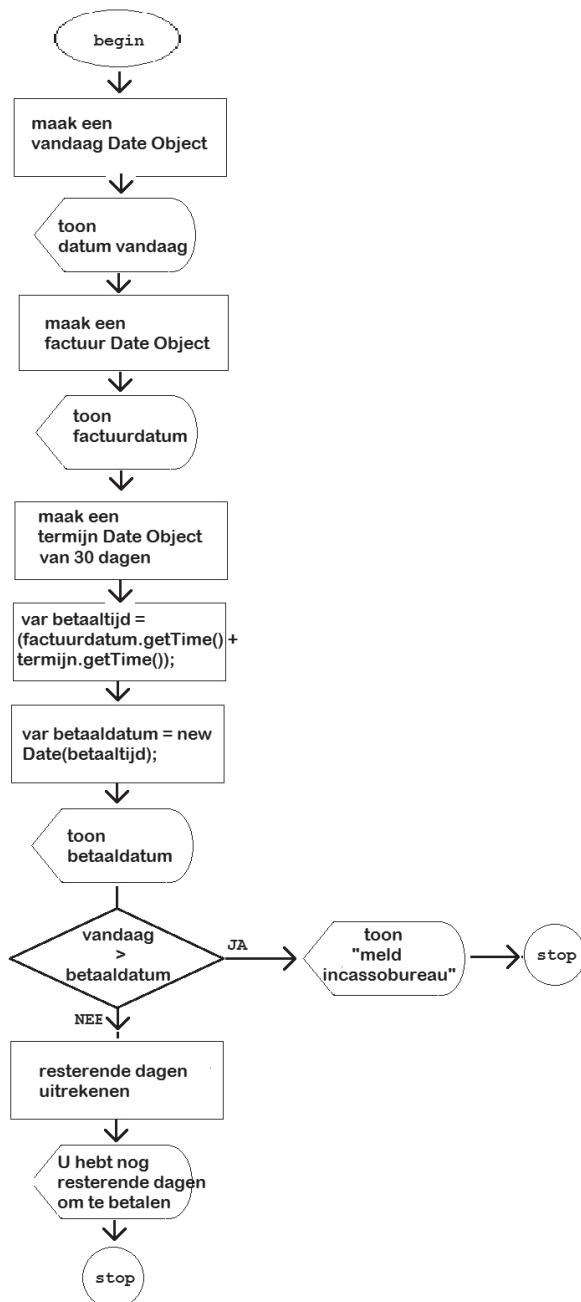
*Als de postcode gelijk is aan “1010ST”, toon je de volgende alertbox:
“Kans op de Nationale Postcode Loterij is 90%.”*

Anders toon je de alertbox:

“Kans op de Nationale Postcode Loterij is 0,069%.”

3.10.10 Web-lab 10

In dit web-lab codeer je een script volgens onderstaand stroomdiagram. Dit script sla je op als **web-lab 10.html**.

Figuur 3.18Stroomdiagram
bij web-lab 10

Afhankelijk van de datum vandaag zal het resultaat er ongeveer zo uitzien:

Web-lab 10

Vandaag is: Mon Oct 28 2013

factuurdatum is: Fri Oct 11 2013

betaaldatum is: Sun Nov 10 2013

U hebt nog: 12 dagen om te betalen

3.10.11 Switch

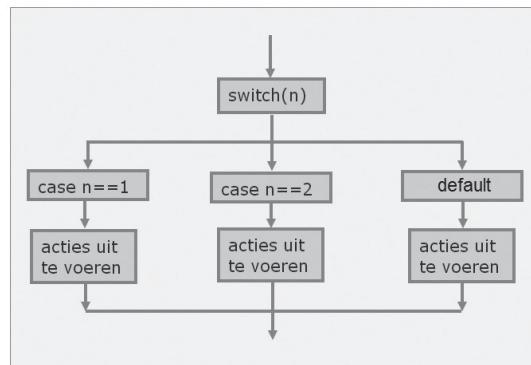
De opdracht `switch` is ook een beslissingsstructuur. Deze kan efficiënter zijn dan de opdracht `if`. We gebruiken de opdracht `switch` om te beslissen welk blok uit veel blokken code moet worden uitgevoerd. De syntaxis van de opdracht `switch` is als volgt:

Syntaxis:

```
switch(n)
{
    case 1:
        acties uit te voeren als n==1
        break;
    case 2:
        acties uit te voeren als n==2
        break;
    default:
        acties uit te voeren als n anders is dan 1 en 2
}
```

Figuur 3.19

Stroomdiagram
van de opdracht
`switch`



- Opgave 53** Open een nieuw bestaand en sla het op als **switch-opdracht.html**. Voeg de volgende code er aan toe.

```
<!DOCTYPE html>
<html lang="nl">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
            charset=UTF-8">
        <title>switch-opdrachten</title>
    </head>
    <body>
        <script type="text/javascript">
            var datum=new Date();
            var vandaag=datum.getDay();
            var uur=datum.getHours();
            document.write("Het is vandaag:" + datum);
            switch(vandaag)
            {
                case 1:
                    document.write(

```

```

        "maandag is mijn eerste stagedag");
        break;
    case 2:
        document.write(
        "dinsdag is mijn tweede stagedag");
        break;
    case 3:
        document.write(
        "woensdag is mijn JavaScriptdag");
        break;
    case 4:
        document.write(
        "donderdag is mijn werkdag");
        break;
    case 5:
        document.write(
        "vrijdag is mijn roostervrije dag");
        break;
    }
}
</script>
</body>
</html>

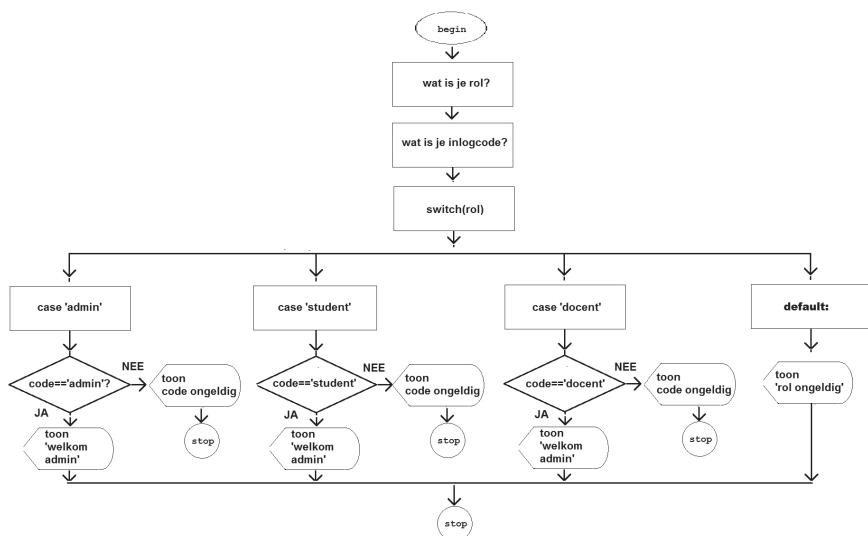
```

3.10.12 Web-lab 11

In dit web-lab codeer je een script volgens onderstaand stroomdiagram.
Dit script sla je op als **web-lab 11.html**.

Figuur 3.20

Stroomdiagram bij
web-lab 11



3.10.13 Web-lab 12

Al je veel beslissingen of opties moet coderen met alleen de if-opdracht, eindig je met if-opdrachten binnen if-opdrachten binnen nog meer if-opdrachten. Dan wordt je code vrijwel onbegrijpelijk. Met de opdracht switch kunnen we op een meer gestructureerde manier beslissingen of opties coderen.

Lab opgave

Maak een nieuw JavaScript-bestand **web-lab 12.html** met de volgende functionaliteit:

1. Vraag de gebruiker om zijn creditcard.

2. Maak een switch-opdracht als volgt (op xxx moet steeds het leenbedrag verschijnen):

a) Case creditcard is **Visa**:

Vraag de gebruiker om een leenbedrag.

Als het leenbedrag groter is dan 800 euro, toon je de volgende alertbox:

Leenbedrag over kredietlimiet van Visa: € 800 !!!

Anders toon je de volgende alertbox:

U hebt xxx euro's geleend!!!

b) Case creditcard is **Mastercard**:

Vraag de gebruiker om een leenbedrag.

Als het leenbedrag groter is dan 1000 euro, toon je de volgende alertbox:

Leenbedrag over kredietlimiet van Mastercard: € 1000 !!!

Anders toon je de volgende alertbox:

U hebt xxx euro's geleend!!!

c) Case creditcard is **ING**:

Vraag de gebruiker om een leenbedrag.

Als het leenbedrag groter is dan 1500 euro, toon je de volgende alertbox:

Leenbedrag over kredietlimiet van ING: € 1500 !!!

Anders toon je de volgende alertbox:

U hebt xxx euro's geleend!!!

d) Als de creditcard geen van bovenstaande kaarten is, toon je de volgende alertbox:

Uw credit card wordt niet geaccepteerd !!!

Tip: Het euroteken coderen we als:

€

3.11 Eigen functies

We hebben kennismoecht met een aantal methodes, bijvoorbeeld de methodes voor het Array-object of het Date-object. Maar we kunnen ook onze eigen functies coderen en hergebruiken. In dit deel van hoofdstuk 3 leren we hoe we onze eigen functies gaan coderen.

3.11.1 Functie of methode

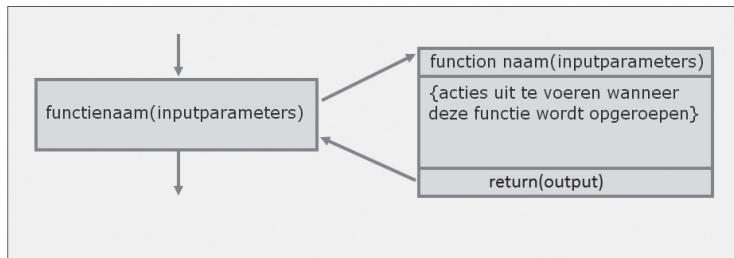
Een methode is een functie binnen een object. Een eigen functie is code die een specifieke taak uitvoert. Eigen functies codeer je apart. Ze zijn dan leesbaarder en je kunt ze hergebruiken wanneer je dezelfde taak moet uitvoeren. In JavaScript declareer je een functie binnen de `<head>`-sectie als volgt:

Syntaxis:

```
function functionnaam ( input parameters )
{
    acties uit te voeren;
    [return (output)] // return is optioneel;
}
```

Figuur 3.21

Stroomdiagram van een functie



Nadat je een functie gecodeerd en gedefinieerd hebt, kun je de functie aanroepen. Een functie roep je als volgt op:

functienaam (input-parameters);

Functie met input

Hier maken we een nieuwe functie met de naam **print** en de parameter **input**. Dit is de signatuur van de functie: **print(input)**.

```
function print(input)
{
    document.writeln(input);
}
```

Dit zou je als volgt kunnen voorstellen: de functie is als een zwarte doos. Je gooi wat input in de zwarte doos (input-parameter). De zwarte doos kan dan iets met de input doen. De acties tussen de accolades worden binnen de zwarte doos uitgevoerd.

Figuur 3.22

Een functie met input

*Functie met input en output*

Hier volgt een voorbeeld van een functie met input en output. Deze functie gebruikt de input om een resultaat te genereren en vervolgens om het resultaat te retourneren:

```
function verdubbelen(input)
{
    output = input*2;
    return(output);
}
```

Figuur 3.23

Een functie met input en output



Een functie aanroepen

Om een functie te gebruiken ('aanroepen') moet je de juiste signatuur van de functie gebruiken. De signatuur van een functie is de functienaam plus de functieparameters binnen haakjes. Deze functie 'verdubbelen' heeft als input een getal nodig of een variabele met een getalwaarde, bijvoorbeeld:

```
verdubbelen(10);
of
verdubbelen(variabele);
```

Figuur 3.24

Een functie aanroepen



Opgave 54 Open een nieuw bestaand en sla het op als **mijnfunctions.html**. Voeg de volgende code er aan toe.

```
<html lang="nl">
<head>
    <meta http-equiv="Content-Type" content="text/html;
        charset=UTF-8">
    <title>mijn functies</title>
    <!--dit script: werkt met functies
        auteur: typ je eigen naam hier
    -->
    <script type="text/javascript">
        function verdubbel( inputgetal )
        {
            outputgetal = inputgetal * 2;
            return(outputgetal);
        }
    </script>
</head>
<body>
    <script type="text/javascript">
        mijngetal = prompt("Typ een getal in");
        resultaat = verdubbel(mijngetal);
        document.writeln("<br />Input-getal:" + mijngetal);
        document.writeln("<br />Output-getal:" + resultaat);
    </script>
</body>
</html>
```

- De functie `verdubbel()` wordt in de JavaScript in `<body>` als volgt opgeroepen: `verdubbel(mijngetal)`
- De input is: `mijngetal`
- De functie `verdubbel()` in de `<head>`-sectie wordt als volgt uitgevoerd:
`function verdubbel(inputgetal)`
De `input-parameter` (`inputgetal`) wordt vervangen door de werkelijk `input: mijngetal`

- De functie voert de volgende acties uit:
`outputgetal = inputgetal * 2`
- De functie geeft als resultaat: `return(outputgetal)`

Het resultaat zou er als volgt kunnen uitzien:

```
Input-getal:44  
Output-getal:88
```

3.11.2 Externe functies

Je codeert een interne functie binnen een HTML-pagina. Maar je kunt ook een externe functie in een extern JavaScript-bestand coderen. Zo kan een groot deel van je programmacode buiten je webpagina blijven.

Een externe functie is ideaal wanneer je herbruikbare code wilt programmeren. Je kunt vanuit je webpagina's de externe functies oproepen. Het externe JavaScript-bestand mag geen HTML-tags hebben. Hieronder zie je een voorbeeld van een extern JavaScript-bestand.

Opgave 55 Zorg ervoor dat **mijnfunctions.js** er als volgt uitziet:

```
function afmelden()  
{  
    var reactie = confirm("Wilt u zich afmelden?");  
    if(reactie == true)  
    {  
        alert("U wordt afgemeld!!!!");  
    }  
}
```

De functie `afmelden()` genereert een confirmbox met de vraag:
"Wilt u zich afmelden?"

Het resultaat van de confirmbox wordt bewaard in de variabele `reactie`.

Als de waarde van de variabele `reactie true` is, wordt een alertbox gegenereerd met de volgende melding: "U wordt afgemeld".

Figuur 3.25

Je herkent een .js-bestand aan het pictogram



mijnfunctions.js

3.11.3 De tag <script source>

Elke webpagina kan gelinkt worden aan een extern .js-bestand door de tag `<script src= >`. Deze tag typ je binnen de `<head>`-sectie.

Opgave 56 Open **mijnfunctions.html** en voer de volgende `<script>`-tags in de `<head>`-sectie en in de `<body>`-sectie eraan toe:

```
<html lang="nl">  
<head>  
    <meta http-equiv="Content-Type" content="text/html;  
        charset=UTF-8">
```

```

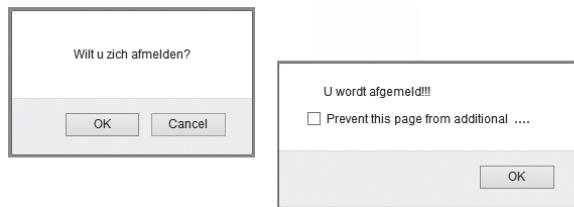
<title>mijn functies</title>
<!--dit script: werkt met functies
    auteur: typ je eigen naam hier
-->
<script type="text/javascript" src="mijnfunctions.js">
<script type="text/javascript">
    function verdubbel( inputgetal )
    {
        outputgetal = inputgetal * 2;
        return(outputgetal);
    }
</script>
</head>
<body>
<script type="text/javascript">
    mijngetal = prompt("Typ een getal in");
    resultaat = verdubbel(mijngetal);
    document.writeln("<br />Input-getal:" + mijngetal);
    document.writeln("<br />Output-getal:" + resultaat);
    afmelden();
</script>
</body>
</html>

```

Hierboven hebben we vanuit de `<head>`-sectie een link naar het externe bestand **mijnfunctions.js** gemaakt. In de `<body>`-sectie hebben we de functie `afmelden()` vanuit het externe script opgeroepen. Het resultaat zie je hieronder.

Figuur 3.26

Het resultaat van
opgave 56



Deze twee boxes worden gegenereerd door de functie `afmelden()` in het externe JavaScript (**mijnfunctions.js**).

Opgave 57 Open **mijnfunctions.js** en voeg de volgende functionaliteit er aan toe:

Maak de nieuwe functie `begroeten()` aan met de volgende functionaliteit:

1. Maak een `Date()`-object dat de systeemdatum genereert.
2. Maak een variabele `uur` met de waarde van de methode `getHours()`.
3. Maak een `if()`-opdracht die het `uur` als volgt test:
Als `uur` tussen 0 t/m 11 is, groet je: Goedemorgen!
Anders als `uur` tussen 12 t/m 17 is, groet je: Goedemiddag!
Anders groet je: Goedenavond!

Open **mijnfunctions.html**. Roep de functie `begroeten()` als volgt aan:

```
begroeten();
```

Het resultaat moet, afhankelijk van de tijd, een van de begroetingen zijn.

```
Input-getal:11  
Output-getal:22  
Goedemorgen!
```

- Opgave 58** Open het externe bestand **mijnfunctions.js** en codeer de volgende functie `koers()` die de huidige eurokoers retourneert.

```
function koers()  
{  
    return(1.36);  
}
```

Open **mijnfunctions.html** Roep de functie `koers()` als volgt aan:

```
document.write('<br />De dollar koers is ' + koers());
```

In dit geval wordt de binnenste functie `koers()` als eerst uitgevoerd. De gereturneerde waarde van de functie `koers()` wordt dan de input naar de buitenste functie `document.write()`. De functie geeft dan 1.36 weer.

```
Input-getal:4  
Output-getal:8  
Goedemorgen!  
De dollar koers is 1.36
```

- Opgave 59** Open het externe bestand **mijnfunctions.js** en codeer de volgende functie `dollar_euro()` die de conversie van dollars naar euro's retourneert.

```
function dollar_euro(dollars)  
{  
    eurokoers = 0.74;  
    euros = dollars * eurokoers;  
    return(euros);  
}
```

Deze functie verwacht een getal als input-parameter. Dat getal represeneert een bedrag in dollars, bijvoorbeeld:

```
dollar_euro(100)
```

De functie moet dit bedrag in euro's converteren.

De functie zegt: als je me een bedrag in dollars geeft, zal ik dat bedrag vermenigvuldigen met de eurokoers en het resultaat retourneren.

Open **mijnfunctions.html**. Roep de functie `dollar_euro()` als volgt aan:

```
document.write('<br />100 dollars is ' +  
dollar_euro(100) +  
' euros\'s');
```

In de tweede regel wordt `dollar _ euro(100)` vervangen door `74`.

In de laatste regel zie je het escapeteken \ binnen een string. Hiermee zeggen we dat het (\')-teken niet het einde van de string is, maar dat het deel uitmaakt van de string.

`'euros\'s'` wordt weergegeven als `euro's`.

Het resultaat moet er als volgt uitzien:

```
Input-getal:100
Output-getal:200
Goedemiddag!
De dollar koers is 1.36
100 dollars is 74 euro's
```

3.11.4 Functies met meerdere parameters

In de volgende opgave coderen we een nieuwe functie `wissel()`. Deze functie heeft twee input-parameters: `bedrag` en `valuta`. De functie moet het `bedrag` naar de aangegeven `valuta` wisselen.

- Opgave 60** Open het externe bestand **mijnfunctions.js** en codeer de volgende functie `wissel()`.

```
function wissel(bedrag, valuta)
{
    dollarkoers = 1.36;
    eurokoers = 0.74;
    if(valuta == 'euro')
    {
        dollars = bedrag * dollarkoers;
        return(dollars);
    }
    else if(valuta == 'dollar')
    {
        euros = bedrag * eurokoers;
        return(euros);
    }
}
```

Open **mijnfunctions.html**. Roep de functie `dollar _ euro()` als volgt aan:

```
document.write('<br />wissel 100 euro\'s in dollars: ' +
wissel(100,'euro'));
document.write('<br />wissel 20 dollars in euro\'s: ' +
wissel(20,'dollar'));
```

Het resultaat moet er als volgt uitzien:

```
Goedemiddag!
De dollar koers is 1.36
100 dollars is 74 euro's
wissel 100 euro's in dollars: 136
wissel 20 dollars in euro's: 14.8
```

3.11.5 Functies dynamisch uitvoeren

In de volgende opgave coderen we een formulier met een inputveld en een knop. In het inputveld gaan we een bedrag intikken. Aan de knop gaan we de functie `dollar_euro()` koppelen, zodat de functie wordt uitgevoerd als je op de knop klikt.

Opgave 61 Open **mijfunctions.html** en voeg er tussen de tags `</script>` en `</body>` het volgende formulier aan toe:

```
<form name='form'>
    <input type="text" name="bedrag" placeholder="bedrag" />
    <input type="button" value="converteren"
    onclick="
        document.write('conversie in euros:' +
        dollar_euro(document.form.bedrag.value))
    "/>
</form>
</body>
</html>
```

Het resultaat moet er als volgt uitzien:

Figuur 3.27

Het resultaat van
opgave 61

```
Goede middag!
De dollar koers is 1.36
100 dollars is 74 euro's
wissel 100 euro's in dollars: 136
wissel 20 dollars in euro's: 14.8
```

3.11.6 Web-lab 13

Maak een nieuw bestand **web-lab 13.html** met de volgende functionaliteit:

1. Maak een formulier zoals te zien is in de volgende figuur:

Figuur 3.28

Formulier in
web-lab 13

vul bedrag in	dollar/euro	converteren
\$		
€		
ruble/euro		
euro/ruble		

2. Koppel de knop ‘converteren’ aan de functie `exchange()` in het externe bestaand **mijfunctions.js**
3. Codeer de functie `exchange()` als volgt:

```
function exchange(bedrag, conversie)
{
}
```

Zorg dat het bedrag wordt geconverteerd uit een van de volgende koersen:

euro_dollarkoers = 1.36
 dollar_eurokoers = 0.74
 euro_roebelkoers = 48.40
 roebel_eurokoers = 0.02

Het resultaat zou bijvoorbeeld het volgende kunnen zijn:

10 roebels voor 0.21 euro's

3.11.7 Web-lab 14

Zorg ervoor dat **web-lab14.html** er als volgt uitziet:

```
<!DOCTYPE html>
<html lang="nl">
<head>
<title>web-lab 14</title>
<!--
dit script: maakt een functie
auteur:
datum:
-->
<script type="text/javascript">
    function vertaal( getal )
    {
        . . .
        return(tekst);
    }
</script>
</head>
<body>
<h3>Vertaal een getal</h3>

. . .

</script>
</body>
</html>
```

De functie `vertaal` moet een inputgetal (1 t/m 10) ‘vertalen’ in tekst, bijvoorbeeld 1 wordt **een** en 10 wordt **tien**. Maak de vertaalfunctie af. In plaats van de twee regels met puntjes moet je de benodigde code toevoegen. Maak in je oplossing gebruik van een array.

Het resultaat moet er ongeveer zo uitzien:

Figuur 3.29

Resultaat van
web-lab14



3.11.8 Globale en lokale variabelen

Een globale variabele is een variabele die zichtbaar is binnen het hele JavaScript. Dat betekent dat een globale variabele overal in je script gewijzigd of gelezen kan worden. Een globale variabele geef je als volgt aan met het prefix var:

```
var variabeleNaam = variabeleWaarde;
```

Een lokale variabele is een variabele binnen een functie. Dat betekent dat deze variabelen buiten de functie ongedefinieerd of niet zichtbaar zijn.

Een lokale variabele geef je binnen een functie als volgt aan zonder het prefix var:

```
variabeleNaam = variabeleWaarde;
```

In de volgende opgave creëren we globale variabelen en binnen een functie ook lokale variabelen.

Opgave 62 Typ het volgende script en sla het op als **globalevars.html**.

```
<!DOCTYPE html>
<html lang="nl">
    <head>
        <title>Globale variabelen</title>
        <meta http-equiv="Content-Type" content="text/html;
            charset=UTF-8">
        <script type="text/javascript">
            <!--dit script: gebruikt globale en lokale variabelen
            -->
            // globale auto vars
            var globale_model = "economie";
            var globale_kleur = "blauw";
            var globale_deuren = 4;
            var globale_radio = true;
            var globale_prijs = 10000;
            function bestelauto(model,kleur,deuren)
            {
                document.write('<br />-----Begin function-----');
                if(typeof arguments[0] === 'undefined')
                {
                    // gebruik globale gegeven
                    model = globale_model;
                }
                if(typeof arguments[1] === 'undefined')
                {
                    // gebruik globale gegeven
                    kleur = globale_kleur;
                }
                if(typeof arguments[2] === 'undefined')
                {
                    // gebruik globale gegeven
                    deuren = globale_deuren;
                }
            }
        </script>
    </head>
    <body>
        <h1>Bestel een auto</h1>
        <form>
            <label>Model:</label>
            <input type="text" value=""/>

```
model
```

</input>
            <label>Kleur:</label>
            <input type="text" value=""/>

```
kleur
```

</input>
            <label>Deuren:</label>
            <input type="text" value=""/>

```
deuren
```

</input>
            <input type="button" value="Bestel" onclick="bestelauto(model,kleur,deuren)"/>
        </form>
    </body>
</html>
```

```

if(deuren == 4)
{
    // gebruik globale gegevens
    prijs = globale_prijs;
    radio = globale_radio;
}
else
{
    // gebruik lokale gegevens
    prijs = 9000;
    radio = false;
}
bestelling = (
    '<br />Model:' + model +
    '<br />Kleur:' + kleur +
    '<br />Deuren:' + deuren +
    '<br />Radio:' + radio +
    '<br />Prijs:' + prijs
);
document.write(bestelling);
    document.write('<br />-----Einde function -----');
}
</script>
</head>
<body>
<h2>Bestel een auto</h2>
<script type="text/javascript">
    bestelauto();
</script>
</body>
</html>

```

In bovenstaand script hebben we globale autogegevens gedeclareerd:

```

// globale auto vars
var globale_model = "economie";
var globale_kleur = "blauw";
var globale_deuren = 4;
var globale_radio = true;
var globale_prijs = 10000;

```

De array arguments[]

De array `arguments[]` is de array binnen een functie waar de input-parameters van de functie worden opgeslagen. Als we een functie definiëren, geven we de signatuur van de functie aan, bijvoorbeeld:

```
function bestelauto(model,kleur,deuren)
```

In dit geval zijn er drie input-parameters: `(model,kleur,deuren)`. Dit betekent dat we maximaal drie input-parameters kunnen doorgeven.

Deze parameters worden in de array `arguments[]` als volgt opgeslagen:

```

arguments[0] = model;
arguments[1] = kleur;
arguments[2] = deuren;

```

Maar binnen de functie bestelauto() hebben we gecontroleerd of er nul, een, twee of drie parameters doorgegeven zijn wanneer we de functie aanroepen:

```
if(typeof arguments[0] === 'undefined')  
{  
    // gebruik globale gegeven  
    model = globale_model;  
}
```

Als de eerste parameter ontbreekt, gebruiken we de globale gegevens voor die parameter. De variabele `model` die we binnen de functie gedeclareerd hebben is een lokale variabele.

Daarna hebben we het volgende getest: als de aantal `deuren` gelijk is aan vier, dan gebruiken we globale gegevens voor `prijs` en `radio`, anders gebruiken we lokale gegevens.

```
if(deuren == 4)  
{  
    // dan globale gegevens  
    prijs = globale_prijs;  
    radio = globale_radio;  
}  
else  
{  
    // lokale gegevens  
    prijs = 9000;  
    radio = false;  
}
```

Als we nu de functie zonder input-parameters aanroepen, krijgen we alleen maar globale gegevens te zien:

```
bestelauto();
```

Het resultaat moet er als volgt uitzien:

```
Bestel een auto  
-----Begin function -----  
Model:economie  
Kleur:blauw  
Deuren:4  
Radio:true  
Prijs:10000  
-----Einde function -----
```

In de volgende opgave kijken naar een globale en een lokale variabele.

Opgave 63 Open `globalevars.html` en voeg de volgende code onder aan het script toe.

```
document.write('<br />De datatype van globale_model '+  
    'typeof globale_model);
```

```
document.write('<br />De datatype van lokale model '+  
    'typeof model); bestelauto();  
</script>  
</body>  
</html>
```

In het resultaat zien we dat buiten de functie de globale variabele gedefinieerd of zichtbaar is, maar dat de lokale variabele ongedefinieerd of onzichtbaar is:

Bestel een auto

```
De datatype van globale_model is: string  
De datatype van lokale model is: undefined  
-----Begin function -----  
Model:economie  
Kleur:blauw  
Deuren:4  
Radio:true  
Prijs:10000  
-----Einde function -----
```

In de volgende opgave kijken we naar de globale en lokale variabelen binnen de functie.

Opgave 64 Open **globalevars.html** en voeg de volgende code binnen en onder aan je functie toe.

```
document.write('<br />De datatype van globale _ model is: ' +  
    'typeof globale _ model );  
document.write('<br />De datatype van lokale model is: ' +  
    'typeof model );  
document.write('<br />-----Einde function -----');
```

In het resultaat zie je dat zowel de globale als de lokale variabelen binnen de functie gedefinieerd en zichtbaar zijn.

```
De datatype van globale_model is: string  
De datatype van lokale model is: undefined  
-----Begin function -----  
Model:economie  
Kleur:blauw  
Deuren:4  
Radio:true  
Prijs:10000  
De datatype van globale_model is: string  
De datatype van lokale model is: string  
-----Einde function -----
```

Functie met input-parameter aanroepen

We gaan nu de functie met de input-parameters `model` als volgt aanroepen:

```
bestelauto('business');
```

- Opgave 65** Open **globalevars.html** en roep de functie met de input-parameter `model` als volgt aan:

```
bestelauto('business');
```

Dan krijgen we de globale gegevens. Maar `model` wordt gespecificeerd door de input-parameter 'business':

```
-----Begin function -----  
Model:business  
Kleur:blauw  
Deuren:4  
Radio:true  
Prijs:10000  
-----Einde function -----
```

We gaan nu de functie met de input-parameters `model` en `kleur` als volgt aanroepen:

```
bestelauto('luxe', 'rood');
```

- Opgave 66** Open **globalevars.html** en roep de functie aan. Geef de input-parameters `model` en `kleur` als volgt door:

```
bestelauto('luxe', 'rood');
```

Dan krijgen we de globale gegevens plus de lokale gegevens `model` en `kleur` te zien:

```
-----Begin function -----  
Model:luxe  
Kleur:rood  
Deuren:4  
Radio:true  
Prijs:10000  
-----Einde function -----
```

We gaan nu de functie met de input-parameters `model`, `kleur` en `deuren` als volgt aanroepen:

```
bestelauto('sport', 'zwart',2);
```

Opgave 67 Open **globalevars.html** en roep de functie aan. Geef de input-parameters `model`, `kleur` en `deuren` als volgt door:

```
bestelauto('sport', 'zwart',2);
```

Dan krijgen we de lokale gegevens `model`, `kleur` en `deuren` te zien.

Maar omdat `deuren` niet 4 is, krijgen we ook de lokale gegevens `radio` en `prijs` te zien. Allemaal lokale variabelen.

```
-----Begin function -----
Model:sport
Kleur:zwart
Deuren:2
Radio:false
Prijs:9000
-----Einde function -----
```

3.12 Lussen

Controlestructuren zoals de if-opdracht, de switch-opdracht en functies helpen ons de volgorde van uitvoering van de instructies in een programma te bepalen. Met lussen kunnen we blokken code herhalen, bijvoorbeeld als je een blok code drie of meer keer wilt laten uitvoeren. Er zijn verschillende lusstructuren. In dit hoofdstuk oefenen we met de volgende lusstructuren.

```
for()-lus
for(in)-lus
while()-lus
do-while()-lus
```

3.12.1 De for()-lus

Met de `for()`-lus kun je een aantal keer hetzelfde blok instructies herhalen. Het aantal keer dat een blok code wordt herhaald, controleren we met een controlevariabele. In de volgende voorbeelden is `x` onze controlevariabele. In de syntaxis van de `for()`-lus zien we twee delen:

Syntaxis:

```
for() is de constructie van de for()-lus
{
    Hier is de body van de for()-lus
}
```

De constructie splitsen we in drie delen:

```
for(deel 1; deel 2; deel 3)
```

Bijvoorbeeld:

```
for(x=0; x<10; x++)
{
    body van de for()-lus
}
```

In deel 1 maken we de controlevariabele, bijvoorbeeld: `x=0`

In deel 2 bepalen we de voorwaarde, bijvoorbeeld: `x < 10`

In deel 3 tellen we de controlevariabele op, bijvoorbeeld: `x++`

In de body van de `for()`-lus coderen we de acties die worden uitgevoerd zolang de voorwaarde `true` is. Bijvoorbeeld:

```
for(x=0;x<6;x++)
{
    document.write(x);
}
```

Hier wordt de actie `document.write(x)` zes keer uitgevoerd en `x` zes keer opgeteld.

Opgave 68 Open een nieuw bestand en sla het op als **for-lus.html**. Codeer het volgende script:

```
<!DOCTYPE html>
<html lang="nl">
    <head>
        <meta http-equiv="Content-Type" content="text/html;
            charset=UTF-8">
        <title>De for-lus</title>
        <!--dit script: werkt met de for-lus
            auteur: typ je eigen naam hier
            -->
    </head>
    <body>
        <script type="text/javascript">
            for (x = 0; x < 3; x++)
            {
                document.writeln("<br />Dit is lus: " + x);
            }
        </script>
    </body>
</html>
```

Het resultaat moet er zo uitzien:

```
Dit is lus: 0
Dit is lus: 1
Dit is lus: 2
```

De `for()`-lus herhaalt de functie `write()` drie keer. We hebben de lus als volgt geconstrueerd:

Deel 1 `x=0` declareert de controlevariabele `x` en zet deze op `0`.

Deel 2 `x<3` is de voorwaarde. Zolang deze voorwaarde `true` is, wordt de lus herhaald. Als de voorwaarde `false` is, wordt de lus beëindigd.

Deel 3 `x++` de variabele `x` wordt bij elke lus opgeteld.

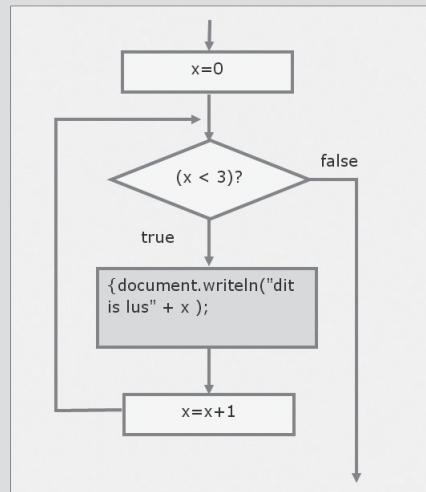
De acties uit te voeren in de body van de lus zijn:

```
document.write( "Dit is lus: " + x );
```

Het stroomdiagram van dit voorbeeld zie je hieronder.

Figuur 3.30

Stroomdiagram
van de for()-lus



Opgave 69 Open **for-lus.html** en voeg de volgende array en lus er aan toe:

```

var weekdag = new Array();
weekdag[0] = "zondag";
weekdag[1] = "maandag";
weekdag[2] = "dinsdag";
weekdag[3] = "woensdag";
weekdag[4] = "donderdag";
weekdag[5] = "vrijdag";
weekdag[6] = "zaterdag";
for(x=0; x<7; x++)
{
    document.write("<br/>Element "+x+" is "+weekdag[x]);
}
  
```

Hier gebruiken we de `for()`-lus om door de elementen van de array te lopen. Het resultaat moet er als volgt uitzien:

```

Element 0 is zondag
Element 1 is maandag
Element 2 is dinsdag
Element 3 is woensdag
Element 4 is donderdag
Element 5 is vrijdag
Element 6 is zaterdag
  
```

We hebben een array `weekdag[]` gemaakt met zeven elementen.

De `for`-lus scant de array `weekdag[]` als volgt:

- Bij de eerste lus krijgt `var x` de waarde 0, dus wordt `weekdag[0]` weergegeven.

- Bij de tweede lus krijgt var x de waarde 1, dus wordt weekdag[1] weergegeven, enzovoort zolang de voorwaarde ($x < 7$) true is.
- Na de zevende lus krijgt var x de waarde 7, dus de voorwaarde ($x < 7$) is false. De for()-lus wordt beëindigd.

Opgave 70 Open **for-lus.html** Codeer een for()-lus zo dat het eerste en het laatste element (zondag en zaterdag), niet worden weergegeven.

3.12.2 De functie charAt()

Kijk eens naar de volgende functie: `charAt(positie)`. Deze functie scant een tekst en geeft als output het teken in positie x van een bepaalde tekst. De input `positie` is een getal, de output is een teken. Een voorbeeld:

```
tekst = "JavaScript";
letter = tekst.charAt(0);
```

De variabele `tekst` kunnen we als volgt voorstellen:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
J	a	v	a	S	c	r	i	p	t

Na het uitvoeren van deze regels heeft de variabele `letter` de waarde 'J'. Dit is het teken in positie 0 van de tekst. Zo kunnen we vanuit alle posities van de tekst het juiste teken halen. Bijvoorbeeld:

```
letter = tekst.charAt(1);
```

Na deze instructie heeft de variabele `letter` de waarde 'a'.

Het volgende script leest een tekst om vervolgens de spaties in de tekst op te tellen. Hiervoor gebruiken we de for()-lus.

Opgave 71 Open **for-lus.html** en voeg de volgende code er aan toe.

```
var tekst = "Hij lijkt een beetje op Nelson Mandela.";
lengte = tekst.length;
spaties = 0;
document.write("<br>");
for(x=0; x<lengte; x++)
{
    teken = tekst.charAt(x);
    document.write(teken);
    if (teken == " ")
    {
        spaties++;
    }
}
document.write("<br>Tekst bevat "+spaties+" spaties");
</script>
</body>
</html>
```

Het resultaat moet er als volgt uitzien:

Hij lijkt een beetje op Nelson Mandela.

Tekst bevat 6 spaties

In dit script wordt de var tekst gedeclareerd met de volgende inhoud:

Hij lijkt een beetje op Nelson Mandela.

De var lengte krijgt de lengte (39) van de inhoud van de variabele tekst en de variabele spaties is (0).

De voorwaarde voor de `for()`-lus is dat `x` kleiner moet zijn dan de lengte van de inhoud van variabele `tekst` dus (`x < lengte`). Zo kan de `for()`-lus de hele tekst scannen.

Je zou de variabele `tekst` als de volgende array kunnen zien:

[0]	[1]	[2]	[3]	[4]	[5]	...
H	i	j		I	i	...

Met behulp van de functie `charAt(x)` kijken we naar het teken bij elke positie `[x]`. Als het teken gelijk is aan een spatie, gaat de spatieteller `spaties` omhoog.

Aan het einde van de lus geef je het resultaat weer.

- Opgave 72** Open **for-lus.html** en voeg de volgende code er aan toe zodat het script de tekens a, e, i, o, u optelt en vervolgens het totale aantal verschillende klinkers in de tekst weergeeft.

Het resultaat moet er als volgt uitzien:

Hij lijkt een beetje op Nelson Mandela.

Aantal a's: 2

Aantal e's: 7

Aantal i's: 2

Aantal o's: 2

Aantal u's: 0

3.12.3 Web-lab 15

Zorg ervoor dat **web-lab 15.html** er als volgt uitziet:

```
<!DOCTYPE html>
<html lang="nl">
  <head>
    <title>web-lab 15</title>
    <!--
      dit script: werkt met de for-lus
      auteur:
      datum:
      -->
  </head>
  <body>
```

```
<script type="text/javascript">
    var getal = new Array();
    getal[0] = "nul";
    getal[1] = "een";
    getal[2] = "twee";
    getal[3] = "drie";
    getal[4] = "vier";
    getal[5] = "vijf";
    getal[6] = "zes";
    getal[7] = "zeven";
    getal[8] = "acht";
    getal[9] = "negen";
</script>
</body>
</html>
```

Lab opgave

Voeg de volgende functionaliteit eraan toe:

Vraag de gebruiker om zijn telefoonnummer.

Geef vervolgens alle cijfers van het telefoonnummer weer als tekst.

Maak hiervoor een for-lus die het telefoonnummer scant.

Gebruik de `getal-array` om alle cijfers van het telefoonnummer te vertalen naar tekst. Het resultaat moet er ongeveer als volgt uitzien:

Figuur 3.31

Web-lab 15



1234567890

Je telefoonnummer is: een twee drie vier vijf zes zeven acht negen nul

Tip: Het telefoonnummer scan je met de for-lus. Gebruik de functie `charAt()` in de for-lus om het juiste getal te vertalen door middel van de `getal-array`.

3.12.4 De for(in)-lus

De `for(in)`-lus gebruiken we om de properties van een object door te lopen. De syntaxis van de `for(in)`-lus is als volgt:

```
for(var property in array)
{
    acties met var property;
}
```

Bij de eerste lus krijgt `property` de naam van de eerste property.

Bij de tweede lus krijgt `property` de naam van de tweede property, enzovoort.

- Opgave 73** Open een nieuw bestand en sla het op als **for-in-lus.html**. Voeg de volgende code er aan toe:

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <meta http-equiv="Content-Type"
          content="text/html;
          charset=UTF-8">
    <title>for(in)-lus</title>
</head>
<body>
    <script>
        var laptops = new Array();
        laptops['macbook'] =
            {'model':'air', 'voorraad':2000, 'prijs':1090};
        laptops['asusbook'] =
            {'model':'wind' , 'voorraad':1000,'prijs':990};
        laptops['dellbook'] =
            {'model':'fire' , 'voorraad':987, 'prijs':890};

        for(var merk in laptops)
        {
            document.write("<br />Merk: " + merk);
        }
    </script>
</body>
</html>
```

Deze `for(in)`-lus loopt door alle properties van de array `laptops[]`. De properties zijn: 'macbook', 'asusbook' en 'dellbook'. Het resultaat moet er als volgt uitzien:

```
Merk: macbook
Merk: asusbook
Merk: dellbook
```

We gaan nu hetzelfde doen met de properties van alle merken.

- Opgave 74** Open **for-in-lus.html** en voeg de volgende `for(in)`-lus binnen de eerste `for(in)`-lus eraan toe:

```
for(var property in laptops[merk])
{
    document.write(
        '<br />' + property + ":" + laptops[merk][property]);
}
```

Hier willen we alle properties van alle laptops zien. Het resultaat moet er als volgt uitzien:

```
Merk: macbook
model:air
voorraad:2000
prijs:1090
Merk: asusbook
model:wind
voorraad:1000
prijs:990
Merk: dellbook
model:fire
voorraad:987
prijs:890
```

3.12.5 Web-lab 16

Zorg ervoor dat **web-lab 16.html** er als volgt uitziet:

```
<!DOCTYPE html>
<html lang="nl">
  <head>
    <title>web-lab 16</title>
    <!--
      dit script: werkt met de for(in)-lus
      auteur:
      datum:
    -->
  </head>
  <body>
    <script type="text/javascript">
    </script>
  </body>
</html>
```

Browsers hebben een eigen object met de naam `navigator`. Codeer nu een `for(in)`-lus die alle properties van het object `navigator` weergeeft.

Het resultaat hieronder laat een aantal van de properties van de browser Firefox zien:

```
mozId : null
mozPermissionSettings : null
mozAlarms : null
mozApps : [xpconnect wrapped (nsISupports, mozIDOMApplicationRegistry,
mozIDOMApplicationRegistry2)]
mozTCPSocket : null
vibrate : function vibrate() { [native code] }
javaEnabled : function javaEnabled() { [native code] }
taintEnabled : function taintEnabled() { [native code] }
requestWakeLock : function requestWakeLock() { [native code] }
mimeTypes : [object MimeTypeArray]
plugins : [object PluginArray]
doNotTrack : unspecified
appCodeName : Mozilla
oscpu : Windows NT 6.3
```

3.12.6 De while()-lus

De `while()`-lus gebruik je net als de `for()`-lus om een blok instructies te herhalen. Of het blok instructies wordt herhaald, bepaal je met een *voorwaarde*. Aan het einde van elke herhaling wordt gecontroleerd of nog steeds aan de voorwaarde wordt voldaan. Als je de lus voortijdig wilt beëindigen, gebruik je de opdracht `break`. Als je een bepaalde lus wilt overslaan, dan gebruik je de opdracht `continue`. De syntaxis van de `while()`-lus is als volgt:

```
while(voorwaarde)
{
    acties;
    // als je een lus wil overslaan:
    [continue;]
    // als je de lus voortijdig wilt beëindigen
    [break;]
}
```

Het volgende script laat een voorbeeld zien van het gebruik van de `while()`-lus:

Opgave 75 Open een nieuw bestand en sla het op als **while-lus.html**. Codeer het volgende script:

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <meta http-equiv="Content-Type" content="text/html;
        charset=UTF-8">
    <title>while-lus</title>
    <!--dit script: werkt met de while-lus
        auteur: typ je eigen naam hier-->
```

```
-->
</head>
<body>
<script type="text/javascript">
    var vandaag = new Date();
    var uur = vandaag.getHours();
    var minuten = vandaag.getMinutes();
    var seconden = vandaag.getSeconds();
    var milliseconden = vandaag.getMilliseconds();
    var lus = milliseconden;
    var finish = lus + 3;
    document.write("<br />while-lus begint om:" +
        uur + ":" + minuten + ":" + seconden + ":" +
        milliseconden);
    while(lus < finish)
    {
        lus = new Date().getMilliseconds();
        document.writeln("<br />Deze lus om:" + lus);
    }
    document.writeln("<br />while-lus eindigt om:" +
        uur + ":" + minuten + ":" + seconden + ":" + lus);
</script>
</body>
</html>
```

Het resultaat is dat de lus loopt gedurende drie milliseconden:

Opgave 76 Open **while-lus.html**. Zorg ervoor dat de while-lus de tweede milliseconde overslaat.

```
if(lus == milliseconden +1)
{
    continue;
}
```

Het resultaat moet er als volgt uitzien. In dit geval wordt milliseconde 848 overgeslagen.

```
while-lus begint om:10:38:36:847
Deze lus om:847
Deze lus om:849
while-lus eindigt om:10:38:36:850
```

Opgave 77 Open **while-lus.html**. Zorg ervoor dat de while-lus voortijdig wordt beëindigd na twee milliseconden.

```
if(lus == milliseconden+2)
{
    document.write("<br />while-lus voortijdig beëindigd");
    break;
}
```

Het resultaat moet er als volgt uitzien. In dit geval wordt de lus op milliseconde 167 doorlopen, milliseconde 168 wordt overgeslagen en op milliseconde 169 wordt de lus voortijdig beëindigd.

```
while-lus begint om:12:40:4:167
Deze lus om:167
Deze lus om:167
Deze lus om:167
Deze lus om:169
while-lus voortijdig beëindigd
while-lus eindigt om:12:40:4:169
```

3.12.7 Web-lab 17

Zorg ervoor dat **web-lab 17.html** er als volgt uitziet:

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <meta http-equiv="Content-Type" content="text/html;
        charset=UTF-8">
    <title>web-lab 17</title>
    <!--dit script: werkt met de while-lus
        auteur: typ je eigen naam hier
        -->
</head>
<body>
    <script type="text/javascript">
        var vragen = new Array();
        var antwoorden = new Array();
        vragen[1] = " 8 X 4 = ? ";
        vragen[2] = " 8 / 4 = ? ";
        vragen[3] = " 16 X 4 = ? ";
        antwoorden[1] = "32";
        antwoorden[2] = "2";
        antwoorden[3] = "64";
        function toets(index)
        {
            // deze functie krijgt een index als input.
            // Met deze index bepaalt de functie de vraag en
            // correct antwoord.
            antwoord = prompt(vragen[index]);
            if(antwoord == antwoorden[index])
            {
                return("Correct!!!")
            }
            else
            {
                return("Fout!!!")
            }
        }
        toets(1);
        toets(3);
        toets(3);
    </script>
</body>
</html>
```

In deze stap vervang je de laatste drie regels:

```
toets(1);
toets(3);
toets(3);
```

door de volgende code:

```
var doorgaan = true;
var index = 0;
```

```

while(doorgaan)
{
    index++;
    resultaat = toets(index);
    doorgaan = confirm(resultaat + " Wil je doorgaan?");
    if(index == 3) break;
}
</script>
</body>
</html>

```

Lab oppgave

Maak de functie `toets()` af. Voeg zeven nieuwe vragen en antwoorden toe. Maak een overzicht van de resultaten dat er ongeveer zo uitziet:

Antwoord 1 was fout

Antwoord 2 was correct

Antwoord 3 was correct

...

Antwoord 10 was correct

Jouw score is ...

3.12.8 De do-while()-lus

De `do-while()`-lus gebruik je net als de `while()`-lus maar deze lus moet gegarandeerd minimaal één lus uitvoeren. In dit geval wordt de `voorwaarde` aan het einde van elke lus gecontroleerd. De syntaxis is als volgt:

```

do {
    acties;
}
while (voorwaarde) ;

```

Opgave 78 Open een nieuw bestand en sla het op als **do-while-lus.html**. Codeer het volgende script:

```

<!DOCTYPE html>
<html lang="nl">
<head>
    <meta http-equiv="Content-Type" content="text/html;
        charset=UTF-8">
    <title>do-while-lus</title>
    <!--dit script: werkt met de while-lus
        auteur: typ je eigen naam hier
        -->
</head>
<body>
    <script type="text/javascript">var lus = 0;
    do {lus++;

        doorgaan = prompt("lus " + lus + "Wil je
            doorgaan?");

    }

```

```
        while (doorgaan) ;  
    </script>  
</body>  
</html>
```

3.13 Het Math-object

Het Math-object is een object anders dan bijvoorbeeld het Date-object. Het Math-object heeft geen constructor, dus we kunnen geen nieuw Math-object construeren. Het is wel een object met statische properties en methodes. Deze statische methodes en properties zijn statisch gedefinieerd in het Math-object en daarom hoeven we geen nieuw object te construeren. In dit hoofdstuk kijken we naar de volgende methodes van het Math-object:

```
abs(x)  
ceil(x)  
floor(x)  
max(x)  
min(x)  
random(x)  
round(x)
```

3.13.1 Math.abs(x)

De methode `abs()` van het Math-object retourneert de absolute waarde van een getal. Bijvoorbeeld:

```
Math.abs('-1');      // retourneert 1  
Math.abs(-2);       // retourneert 2  
Math.abs(null);     // retourneert 0  
Math.abs("string"); // retourneert NaN  
Math.abs();          // retourneert NaN
```

3.13.2 Math.ceil(x)

De methode `ceil()` retourneert de plafondwaarde van een getal (dat is de kleinste integer gelijk aan of groter dan het getal), bijvoorbeeld:

```
Math.ceil(3);        // retourneert 3  
Math.ceil(3.3);     // retourneert 4  
Math.ceil(3.8);     // retourneert 4  
Math.ceil(4.5);     // retourneert 5  
Math.ceil(5.004);   // retourneert 6
```

3.13.3 Math.floor(x)

De methode `floor()` retourneert de vloerwaarde van een getal (dat is de grootste integer gelijk aan of kleiner dan het getal), bijvoorbeeld:

```
Math.floor(3);       // retourneert 3  
Math.floor(3.3);     // retourneert 3  
Math.floor(3.8);     // retourneert 3  
Math.floor(4.5);     // retourneert 4  
Math.floor(5.004);   // retourneert 5
```

3.13.4 Math.max()

De methode `max()` retourneert het hoogste getal uit een groep getallen, bijvoorbeeld:

```
Math.max(3,6,8,5,3,2); // retourneert 8
```

- Opgave 79** Open een nieuw bestand en sla het op als **math-object.html**. Voeg de volgende code er aan toe:

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <meta http-equiv="Content-Type" content="text/html;
        charset=UTF-8">
    <title>Math-Object</title>
    <!--dit script: werkt met het Math-Object
        auteur: typ je eigen naam hier
        -->
</head>
<body>
    <script type="text/javascript">
        var mijnArray = new Array(3.51,3.50,3.49,-3.55,-3.50,-3.49);
        document.write('<br /> Hoogste element in mijnArray is: '+
            Math.max.apply(null, mijnArray));
    </script>
</body>
</html>
```

Het resultaat zie je hieronder:

```
Hoogste element in mijnArray is: 3.51
```

3.13.5 Math.min()

De methode `min()` retourneert het laagste getal uit een groep getallen, bijvoorbeeld:

```
Math.min(3,6,8,5,3,2); // retourneert 2
```

- Opgave 80** Open **math-object.html** en voeg de volgende code er aan toe:

```
document.write('<br /> Laagste element in mijnArray is: '+
    Math.min.apply(null, mijnArray));
</script>
</body>
</html>
```

Het resultaat zie je hieronder:

```
Hoogste element in mijnArray is: 3.51
Laagste element in mijnArray is: -3.55
```

3.13.6 Math.round()

De methode `round()` retourneert de afgeronde integer waarde van het getal, bijvoorbeeld:

```
Math.round(3.51);      // retourneert 4
Math.round(3.50);      // retourneert 4
Math.round(3.49);      // retourneert 3
Math.round(-3.55);     // retourneert -4
Math.round(-3.50);     // retourneert -3
Math.round(-3.49);     // retourneert -3
```

Opgave 81 Open **math-object.html** en voeg de volgende code er aan toe:

```
for(i=0; i < mijnArray.length; i++)
{
    mijnArray[i] = Math.round(mijnArray[i]);
}
document.write('<br />Afgeronde elementen: ' +
mijnArray);
</script>
</body>
</html>
```

Het resultaat zie je hieronder:

```
Hoogste element in mijnArray is: 3.51
Laagste element in mijnArray is: -3.55
Afgeronde elementen: 4,4,3,-4,-3,-3
```

3.13.7 Math.random()

De methode `random()` retourneert elke keer weer een willekeurige waarde tussen (0 en 1) inclusief 0, exclusief 1, bijvoorbeeld:

```
Math.random();
// retourneert bijvoorbeeld de willekeurige waarde
0.137040278683634

Math.random();
// retourneert een nieuwe willekeurig waarde 0.837040278683634
```

Opgave 82 Open **math-object.html** en voeg de volgende code er aan toe:

```
document.write(
'<br />Vul mijnArray met willekeurige getallen');
for(x=0;x < mijnArray.length;x++)
{
    mijnArray[x] = Math.floor((Math.random()*10));
    document.write('<br />' + mijnArray[x]);
}
</script>
</body>
</html>
```

Het resultaat moet zo iets zijn als:

Vul mijnArray met willekeurige getallen

1
0
2
6
9
3

3.13.8 Web-lab 18

In web-lab 18 maak je eerst de array `alfa = ['a','b','c', etc.]`. Daarna codeer je de functie `nieuwewoorden()` die nieuwe woorden van vijf willekeurige letters verzint. Je zult combinaties zoals die hieronder zien:

zelqv
jgifa
zuaky

3.14 String-methodes

Je zou een tekst- of stringvariabele als een array kunnen zien. Het eerste teken van de tekst is dan element [0], het tweede teken is dan element [1], enzovoort. JavaScript heeft een aantal ingebouwde methodes voor strings. In dit hoofdstuk kijken we naar de volgende string-methodes:

`indexOf()`
`length`
`charAt()`
`split()`
`substring()`
`substr()`
`toLowerCase()`
`toUpperCase()`

3.14.1 indexOf()

Een string is een tekenreeks oftewel een tekst. Elk teken heeft een index die de positie in de tekst aangeeft. De eerste positie is de nul-positie. De methode `indexOf()` geeft de positie van een bepaald teken in de tekst. Als de letter niet gevonden wordt, is het resultaat -1.

```
string = "mijn tekst"  
string.indexOf("k")      // retourneert 7  
string.indexOf("a")      // retourneert -1
```

Opgave 83 Open een nieuw bestand en sla het op als **string-methodes.html**. Codeer het volgende script:

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <meta http-equiv="Content-Type" content="text/html;
    charset=UTF-8">
    <title>string-methodes</title>
    <!--dit script: gebruikt string methodes -->
</head>
<body>
    <script type="text/javascript">
        var title = "javascript";
        document.write("<br />Index van letter s is:" +
        title.indexOf("s") );
    </script>
</body>
</html>
```

In dit voorbeeld geeft de methode `title.indexOf("s")` de positie van de letter “s” in de titel “javascript”, dus het getal 4.

Het resultaat zie je hieronder:

Index van letter s is: 4

We kunnen ook de positie van een woord in een tekst vinden. Dit doen we in de volgende opgave.

Opgave 84 Open **string-methodes.html** en voeg de volgende code er aan toe:

```
var string =
"dit is een lange tekst met daarin het woord javascript";
document.write("<br />Index van het woord javascript is:" +
string.indexOf("javascript"));
document.write("<br />Index van het woord HTML is:" +
string.indexOf("HTML"));
```

In dit voorbeeld geeft de methode `tekst.indexOf("javascript")` de beginpositie van het woord **javascript** in de tekst, dus **43**. Als het teken of woord niet gevonden wordt, dan geeft de methode de waarde -1 terug.

Het resultaat van de opgave zie je hieronder:

Index van letter s is:4
Index van het woord javascript is:44
Index van het woord HTML is:-1

In de volgende opgave gaan we testen wat de naam van de actieve browser is. We kijken in de property `userAgent` van het object `navigator` en zoeken naar de volgende drie woorden:

MSIE
Firefox
Chrome

Opgave 85 Open **string-methodes.html** en voeg de volgende code er aan toe:

```
if(navigator.userAgent.indexOf("MSIE") != -1)
{
    document.write("<br />Browser is Internet Explorer");
}
else if(navigator.userAgent.indexOf("Firefox",0) != -1)
{
    document.write("<br />Browser is Firefox");
}
else if(navigator.userAgent.indexOf("Chrome") != -1)
{
    document.write("<br />Browser is Google Chrome");
}
```

Het resultaat van de opgave uitgevoerd met de Firefox-browser zie je hieronder:

```
Index van letter s is:4
Index van het woord javascript is:43
Index van het woord HTML is:-1
Browser is Firefox
```

In dit voorbeeld geeft de methode `indexOf("MSIE")` de positie van het woord **MSIE** in de property `userAgent` van het object `navigator`. Als de positie `-1` is, betekent dit dat MSIE niet werd gevonden en ook dat de browser niet Internet Explorer is. Op dezelfde manier hebben we getest op de browsers Firefox en Chrome.

3.14.2 Length

De property `length` geeft de lengte van een string terug. In de volgende opgave gebruiken we de property `length` van een string-object.

```
gebruiker = "supergebruiker";
gebruiker.length // retourneert 14
```

Opgave 86 Open **string-methodes.html** en voeg de volgende code er aan toe:

```
var title = "JavaScript";
document.write("<br />De lengte van title is: " +
title.length);
```

In dit voorbeeld geeft de property `length` de lengte van de string "JavaScript" weer.

De lengte van title is: 10

3.14.3 `charAt()`

De methode `charAt()` geeft het teken op een bepaalde positie in de string terug.

```
gebruiker = "supergebruiker";
gebruiker.charAt(4) // retourneert r
```

Opgave 87 Open [string-methodes.html](#) en voeg de volgende code er aan toe:

```
var title = "JavaScript";
var spiegeling = " ";
for(x=0; x < title.length; x++)
{
    teken = title.charAt(x);
    spiegeling = teken + spiegeling;
}
document.write("<br />JavaScript gespiegeld: " + spiegeling);
```

In dit voorbeeld geeft de methode `charAt(x)` het teken op positie *x*. De for-lus leest elk teken en maakt een nieuwe string in omgekeerde volgorde.

De lengte van title is: 10

JavaScript gespiegeld: tpircSavaJ

3.14.4 `split()`

Met de methode `split()` kun je een lange string splitsen in kleinere strings. De splitsing is gebaseerd op een teken, bijvoorbeeld een spatie of komma's. De gesplitste strings worden in een array geplaatst.

```
namen = "Jozef, Jan, Umut";
array = namen.split(",");
// retourneert de volgende array:
array[0] = "Jozef";
array[1] = "Jan";
array[2] = "Umut";
```

Opgave 88 Open [string-methodes.html](#) en voeg de volgende code er aan toe:

```
var tekst = "dit is een lange tekst " +
"met daarin het woord javascript";
splitsing = tekst.split(" ");
document.write("<br />De splitsing array is: " + splitsing);
```

In dit voorbeeld geeft de methode `split()` de volgende array terug:

```
splitsing[0] = "dit";
splitsing[1] = "is";
splitsing[2] = "een";
splitsing[3] = "lange";
splitsing[4] = "tekst";
splitsing[5] = "met";
splitsing[6] = "daarin";
splitsing[7] = "het";
splitsing[8] = "woord";
splitsing[9] = "javascript";
```

De lengte van title is: 10

JavaScript gespiegeld: tpircSavaJ

De splitsing array is: dit,is,een,lange,tekst,met,daarin,het,woord,javascript

3.14.5 `substring()`

De methode `substring()` geeft een deel van een string terug. De `substring` is gebaseerd op een *begin-index* en een *eind-index*, bijvoorbeeld

```
string = "abcdefg"
string.substring(3,6) // retourneert "defg"
```

Het resultaat hier is de substring die begint met begin-index [3] en eindigt met eind-index [6].

Opgave 89 Open `string-methodes.html` en voeg de volgende code er aan toe:

```
var string = "dit is een lange string " +
"met daarin het woord javascript";
document.write(
"<br />Substring van string: " + string.substring(10,23));
```

In deze opgave geeft de methode `substring(10,23)` de substring met de tekst `lange string` terug.

De lengte van title is: 10

JavaScript gespiegeld: tpircSavaJ

De splitsing array is: dit,is,een,lange,string,met,daarin,het,woord,javascript

Substring van string: lange string

3.14.6 `substr()`

De methode `substr()` geeft ook een deel van een string terug. De `substr` is gebaseerd op een *begin-index* en een *lengte*, bijvoorbeeld:

```
gebruiker = "supergebruiker"
gebruiker.substr(5,9) // retourneert "gebruiker";
```

De methode geeft de substring terug beginnend met index [5] en een lengte van 9.

In de volgende opgave gebruiken we de methode `substr` om het woord `javascript` uit de string te verwijderen.

Opgave 90 Open **string-methodes.html** en voeg de volgende code er aan toe:

```
string =  
string.substr( 0, string.indexOf("javascript")) +  
string.substr( string.indexOf("javascript") + 10, string.  
length);  
document.write("<br />String is nu: " + string);
```

In dit geval gebruik je de methode `indexOf()` voor het bepalen van de beginpositie van het woord `javascript`. Het resultaat zie je hieronder:

De splitsing array: dit,is,een,lange,string,met,daarin,het,woord,javascript
Substring van string: lange string
String is nu: dit is een lange string met daarin het woord

3.14.7 `toLowerCase()` en `toUpperCase()`

De methodes `toLowerCase()` en `toUpperCase()` veranderen een string in hoofdletters of in kleine letters, bijvoorbeeld:

```
naam = "Marlies"  
naam.toLowerCase()           // retourneert marlies  
naam.toUpperCase()          // retourneert MARLIES
```

Opgave 91 Open **string-methodes.html** en voeg de volgende code er aan toe:

```
var cursus = prompt("Kies een van de volgende"+  
"cursussen HTML of JavaScript");  
if (cursus.toLowerCase() == "html" ||  
cursus.toLowerCase() == "javascript")  
{  
    document.write("<br />Je hebt: " + cursus.toUpperCase() +  
    " gekozen.");  
}  
else  
{  
    document.write("<br />Kies een geldige cursus");  
}
```

In deze opgave gebruiken we de methode `toLowerCase()` om de invoer van de gebruiker te wijzigen in kleine letters. De gebruiker mag dan **JavaScript** of **javascript** of **JAVAScript** of **HTML** of **Html** intypen, maar `cursus.toUpperCase()` zet de invoer altijd om in hoofdletters, **JAVASCIPT** of **HTML**.

3.14.8 Web-lab 19

Bij **web-lab 19.html** codeer je de functie `vervangen(doc, woord, nieuwewoord)` die een woord in een tekst zoekt en vervangt door het nieuwewoord.

```
doc = "Dit document is een lang document maar ook een  
simpel document";  
function vervangen(docum,woord,nieuwewoord)  
{  
    . . . . .  
}  
  
doc = vervangen(doc,'document','script');  
document.write(doc);
```

Op de puntjes codeer je de functie. Het resultaat moet als volgt uitzien:

Dit script is een lang script maar ook een simpel script

3.15 Algoritmes

Een algoritme is:

- een set instructies of een lijst van stappen in een bepaalde volgorde...
- binnen een computerprogramma...
- om een specifiek probleem op te lossen.

Deze problemen kunnen van simpel tot zeer complex zijn. Dankzij algoritmes nemen computers besluiten voor ons. Goede algoritmes vormen vaak de basis van een efficiënt computerprogramma. Een klassiek voorbeeld van een algoritme in de informatica is het algoritme binary-search.

3.15.1 Het algoritme binary-search

Het algoritme binary-search is een algoritme dat nauwkeurig de stappen beschrijft die uitgevoerd moeten worden om een zoekopdracht op te lossen.

In de informatica vindt binary-search de positie van een specifieke input-waarde (de zoeksleutel) binnen een geïndexeerde array [0][1][2]... Het algoritme werkt als volgt:

Stap 1: Het algoritme vergelijkt de zoeksleutel met de middel-index in de array.

Stap 2: Als de zoeksleutel en de middel-index overeenkomen, dan is het element gevonden en de waarde gereturneerd.

Stap 3: Als de zoeksleutel kleiner is dan de middel-index dan stel je de array gelijk aan de eerste helft van je array.

Herhaal stap 1 met de eerste helft van je array.

Stap 4: Als de zoeksleutel groter is dan de middel-index dan set je de array gelijk aan de tweede helft van je array.

Herhaal stap 1 met de tweede helft van je array.

Stap 5: Als de array leeg is wordt ‘Sleutel niet gevonden’ gereturneerd.

Een binary-search halveert het aantal te doorzoeken elementen bij elke herhaling, zodat het vinden van een element veel minder tijd kost.

3.15.2 Een voorbeeld

Stel je voor dat je op zoek bent naar sleutel 16 in een array. Dan moet het algoritme de volgende stappen uitvoeren (zie figuur 3.32):

Stap 1: Vergelijk de sleutel [16] met de middel-index .

Stap 2: Als de sleutels overeenkomen, dan is het element gevonden en wordt de waarde geretourneerd.

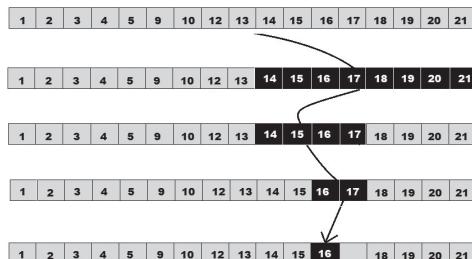
Stap 3: Als de sleutel kleiner is dan de middel-index dan geldt:
sub-array = eerste helft van je array
Herhaal stap 1 met de sub-array (eerste helft).

Stap 4: Als de sleutel groter is dan de middel-index dan geldt:
sub-array = tweede helft van je array
Herhaal stap 1 met de sub-array (tweede helft).

Stap 5: Als de sub-array leeg is, dan wordt ‘Sleutel niet gevonden’ geretourneerd.

In de volgende figuur zie je dat het algoritme 4 keer wordt herhaald voordat de sleutel [16] gevonden is.

Figuur 3.32



3.15.3 Pseudocode

Als je een complex algoritme gaat coderen, is het soms handig om eerst met pseudocode te beginnen. Pseudocode is nog geen programma, maar geeft aan in grote lijnen wat moet gebeuren. Hier volgt een voorbeeld van een pseudocode voor het algoritme binary-search:

```
BinarySearch(array, key, beginIndex, eindIndex)
if beginIndex > eindIndex
then
    return NIET _ GEVONDEN
else
    middenIndex = (beginIndex+eindIndex) / 2
    if key = middenIndex
    then
        return array[middenIndex]
    else if key < middenIndex
    then
        return BinarySearch(array, key, beginIndex,
middenIndex -1)
    else
        return BinarySearch(array, key, middenIndex+1,
eindIndex)
```

Pseudocode kunnen we naar alle programmeertalen vertalen. In de volgende opgave vertalen we de pseudocode naar JavaScript.

Opgave 92 Open een nieuw bestand en sla het op als **binary-search.html**. Voeg de volgende code er aan toe:

```
<!DOCTYPE html>
<html lang="nl">
    <head>
        <meta http-equiv="Content-Type"
              content="text/html;
              charset=UTF-8">
        <title>Binary-search</title>
    </head>
    <body>
        <script>

            var studenten = new Array(16);
            studenten[0] = {'naam':'Navajo', '06':'1234567800'};
            studenten[1] = {'naam':'Zakaria', '06':'1234567801'};
            studenten[2] = {'naam':'Hamsa', '06':'1234567802'};
            studenten[3] = {'naam':'Guus', '06':'1234567803'};
            studenten[4] = {'naam':'Mitchel', '06':'1234567804'};
            studenten[5] = {'naam':'Jan', '06':'1234567805'};
            studenten[6] = {'naam':'Idris', '06':'1234567806'};
            studenten[7] = {'naam':'Kees', '06':'1234567807'};
            studenten[8] = {'naam':'Umut', '06':'1234567808'};
            studenten[9] = {'naam':'Charles', '06':'1234567809'};
            studenten[10] = {'naam':'Pedro', '06':'1234567810'};
            studenten[11] = {'naam':'Joshua', '06':'1234567811'};
            studenten[12] = {'naam':'Angelo', '06':'1234567812'};
            studenten[13] = {'naam':'Peter', '06':'1234567813'};
            studenten[14] = {'naam':'Audi', '06':'1234567814'};
            studenten[15] = {'naam':'Arnold', '06':'1234567815'};

            function zoekalgoritme(studenten, key, imin, imax)
            {
                // test of array leeg is
                if (imax < imin)
                {
                    // array is leeg
                    return('Niet gevonden');
                }
                else
                {
                    imid= Math.ceil(imin + ((imax - imin) / 2));
                    // vergelijken
                    if (imid > key)
                    {
                        // zoek in onderste sub-array
                        document.write('<br />onderste imin:' + imin);
                        document.write('<br />onderste imid:' + imid);
                        return(zoekalgoritme(studenten, key, imin, imid-1));
                    }
                }
            }
        </script>
    </body>
</html>
```

```
        else if (imid < key)
        {
            // zoek in bovenste sub-array
            document.write('<br />bovenste imax:' + imax);
            document.write('<br />bovenste imid:' + imid);
            return(zoekalgoritme(studenten, key, imid+1, imax));
        }
        else
        {
            // key is gevonden
            return(studenten[imid]['naam']+studenten[imid]['06']);
        }
    }
}

var id = parseInt(prompt('Zoek student ID'));
var eindindex = studenten.length;
document.write('eindindex:' + eindindex + '<br />');
var beginindex = 0;

var student=zoekalgoritme(studenten,id,beginindex,eindindex);
document.write('<br />Student gevonden:' + student);
</script>
</body>
</html>
```

3.15.4 Web-lab 20

Schrijf een algoritme dat een piramide weergeeft. Als bijvoorbeeld het inputgetal 9 is, dan genereert het algoritme een piramide zoals hieronder.

```
0
01
012
0123
01234
012345
0123456
01234567
012345678
```

Hieronder schrijven we de pseudocode voor het algoritme.

1. Typ array lengte in.
2. Lees lengte.
3. Maak array(lengte).
4. Array invullen als volgt:
for(i=0; i<=lengte; i++) array[i]=i
5. Voor elke rij
 write elementen [0] t/m [rij - 1]

Open een nieuw bestand en sla het op als **web-lab 20.html**. Schrijf het volgende script dat gebaseerd is op bovenstaand algoritme.

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <meta http-equiv="Content-Type" content="text/html;
        charset=UTF-8">
    <title>web-lab 20</title>
</head>
<body>
    <script>
        var lengte = parseInt(prompt("Typ een getal in:"));
        piramide = new Array(lengte);
        for (var i = 0; i <=lengte; i++)
        {
            piramide[i]= i;
        }
        document.write(piramide);
        for (var rij=1; rij <= lengte; rij++)
        {
            document.write("<br />");
            for (var i = 0; i < rij; i++)
            {
                document.write(piramide[i]);
            }
        }
    </script>
</body>
</html>
```

3.15.5 Web-lab 21

Schrijf een algoritme dat een piramide weergeeft. Als bijvoorbeeld het inputgetal 10 is, dan genereert het algoritme een piramide zoals hieronder.

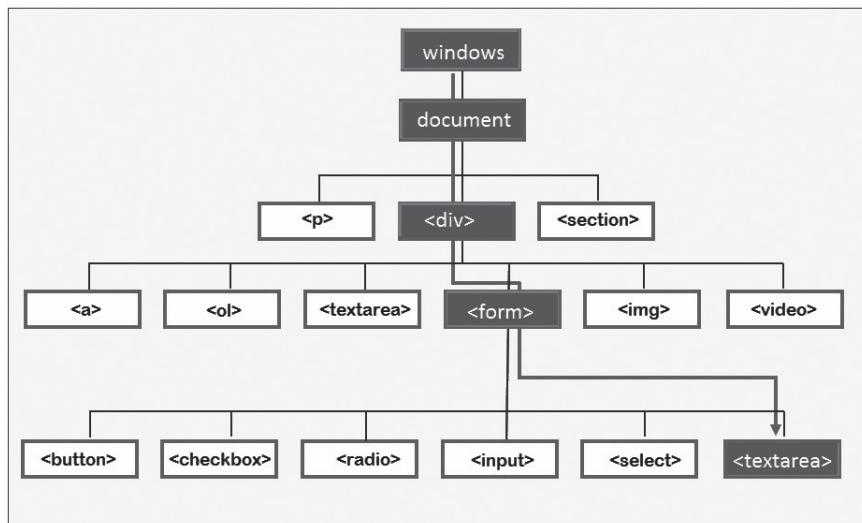
```
0123456789
012345678
01234567
0123456
012345
01234
0123
012
01
0
```

3.16 Het Document Object Model

Het *Document Object Model* (afgekort tot DOM) is een model voor het benaderen van gestructureerde documenten zoals een HTML-document. Met JavaScript kun je alle objecten die geassocieerd zijn met je webpagina benaderen en manipuleren. Om een bepaald object te benaderen, moet je een documenthiërarchie doorlopen (zie figuur 3.33).

Figuur 3.33

Hiërarchie van objecten in een formulier



Om bijvoorbeeld de textarea (tekstveld) in figuur 3.33 te benaderen, loop je de volgende hiërarchie door:

```
window.document.div.form.textarea
```

3.16.1 getElementById

Een andere manier om HTML-elementen vanuit JavaScript te benaderen is via het id-attribuut van het element. Dat doen we met:

```
getElementById()
```

Met de opdracht `getElementById` kun je de attributen van een element wijzigen. Je hebt bijvoorbeeld een ``-element als volgt in je webpagina gecodeerd:

```

```

Dan kun je het element met `getElementById` als volgt manipuleren:

```
document.getElementById("image").src = "image.jpg";
```

Eerst benaderen we het ``-element door gebruik te maken van het id-attribuut `id="image"`. Daarna wijzigen we het `src`-attribuut. Hiermee hebben we het ``-element een nieuwe afbeelding toegewezen. Het was `src="afbeelding.jpg"` en het is `src = "image.jpg"` geworden.

3.16.2 DOM-events (gebeurtenis)

Een voorbeeld van een event is wanneer door de gebruiker op een knop wordt geklikt, of als de waarde van een veld verandert. In JavaScript kunnen we acties programmeren en deze acties kunnen we koppelen aan events. Zodra een bepaalde event zich voordoet, worden de gekoppelde acties uitgevoerd. Hieronder zie je een lijst met de verschillende DOM-events:

- window-events
- form-events
- keyboard-events
- mouse-events

In de volgende paragrafen oefenen we met de verschillende DOM-events. We beginnen met de window-events.

3.16.3 Window-events

Window-events zijn events op het niveau van het browservenster of het document. Hieronder zien we de event-naam en wanneer de event wordt getriggerd of geactiveerd.

Events	Gebeurt wanneer
onload	het document wordt geladen
onunload	het document wordt ontladen

Je vangt window-events op in <body>-elementen. Bijvoorbeeld:

```
<body onload="alert('JavaScript wordt geladen')">
```

In de volgende opgave coderen we de functie `showklok()` en koppelen we de functie aan de `onload`-event:

```
<body onload="showklok();">
```

Opgave 93 Open een nieuw bestand en sla het op als **dom-events.html**. Voeg de volgende code er aan toe:

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <meta http-equiv="Content-Type"
          content="text/html;
          charset=UTF-8">
    <title>DOM events</title>
    <script type="text/javascript" src="dom-events.js"></script>
    <script type="text/javascript">

        document.write(
            '<form name="klok"> ' +
            '<input type="text" ' +
            'name="display" ' +
            'size="8" ' +
            'style="background-color: #30CE28; font-size:44px" >' +
            '</form>');

        var nu = new Date();
        var uur = nu.getHours();
        var min = nu.getMinutes();

        var alterneren = 0;
        function showklok()
        {
```

```

min=((min < 10) ? "0" : "") + min;
if (alterneren)
{
    document.klok.display.value=uur + ":" + min;
    alterneren = 0;
}
else
{
    document.klok.display.value=uur + " " + min;
    alterneren = 1;
}

nu = new Date();
uur = nu.getHours();
min = nu.getMinutes();
setTimeout("showklok()",1000);
}
</script>
</head>
<body onload="showklok();">
</body>
</html>

```

3.16.4 De functie setTimeout()

De functie `setTimeout()` voert steeds, na een aangegeven aantal milliseconden, een actie (of je eigen functie) uit. Deze functie heeft twee input-parameters nodig: de functienaam en de aantal milliseconden voor het herhaaldeelijke uitvoeren van de functie:

```
setTimeout("functienaam",milliseconden);
```

In bovenstaande opgave hebben we de functie `showklok()` getriggerd met de `onload`-event van het `<body>`-element. Daarna hebben we, binnen deze functie `showklok()`, dezelfde functie iedere seconde getriggerd.

Het resultaat van deze opgave zie je hieronder.

Figuur 3.34

Resultaat van de functie `showklok()`

11:30

3.16.5 De functie clearTimeout()

Om een timerfunctie te stoppen, gebruiken we de functie `clearTimeout()`:

```
clearTimeout(functienaam);
```

- Opgave 94** Open **dom-events.html** en zorg ervoor dat je klok ook seconden weer geeft. Het resultaat moet er als volgt uitzien:

Figuur 3.35

Resultaat van opgave 94

11:38:05

Opgave 95 Open **dom-events.html** en voeg het volgende `<div>`-element er aan toe:

```

<!DOCTYPE html>
<html lang="nl">
<head>
    <meta http-equiv="Content-Type"
          content="text/html;
          charset=UTF-8">
    <title>Web-lab 22</title>
    <!--dit script: geeft afbeeldingen weer in willekeurige
        volgorde -->
    <script type="text/javascript" src="dom-events.js"></script>
</head>
<body onLoad="showMedia();">
    <div>
        <a href="Javascript:showMedia(randomIndex);">
            
        </a>
    </div>
</body>
</html>

```

Opgave 96 Open een nieuw bestand en sla het op als **dom-events.js**. Voeg de volgende code er aan toe:

```

// Globale variabelen
var afbeeldingen = new Array (
    "galaxy-s4.png",
    "ipad.jpg",
    "laptop.jpg");
var max = afbeeldingen.length;
var randomIndex=0;
function showMedia()
{
    randomIndex = Math.floor((Math.random() * max));
    document.getElementById("media").src =
    afbeeldingen[randomIndex];
}

```

Het resultaat moet er als volgt uitzien:

Figuur 3.36

Resultaat van
opgave 96



Opgave 97 Open **dom-events.js** en zorg ervoor dat de functie `showMedia()` zichzelf om de 500 milliseconden herhaalt. Het resultaat moet zijn dat na een klik op de afbeelding om de 500 milliseconden een nieuwe afbeelding wordt getoond.

3.16.6 Mouse-events

Je vangt een mouse-event op tijdens het bewegen van de muis.
Bijvoorbeeld:

```

```

Hieronder volgt een tabel met de meeste voorkomende mouse-events:

Events	Gebeurt wanneer
onclick	op een form-element wordt geklikt
onmouseup	de muisknop wordt losgelaten
onmousedown	de muisknop wordt ingedrukt
onmousemove	de muis beweegt
onmouseout	de muis beweegt weg van een element
onmouseover	de muis beweegt over een element

In de volgende opgave vangen we de events `mouseover` en `mouseout` op met acties op een afbeelding.

Opgave 98 Open **dom-events.html** en voeg de volgende mouse-events er aan toe:

```

```

Het resultaat moet zijn dat wanneer je met de muis over de afbeelding beweegt de afbeelding groter wordt; als je met de muis van de afbeelding af beweegt wordt de afbeelding kleiner.

3.16.7 Form-events

Form-events zijn events die plaatsvinden binnen formulierelementen.
Hieronder volgt een tabel met de meeste voorkomende form-events:

Events	Gebeurt wanneer
onchange	een element is gewijzigd
onblur	een element raakt buiten focus
onfocus	een element heeft de focus
onselect	een element is geselecteerd
onsubmit	het formulier is verstuurd
onreset	het formulier is gereset

Bij de volgende opgaven gaan we onderstaand formulier gebruiken. We zullen zien hoe we ieder ingevulde waarde in het formulier kunnen benaderen en verwerken.

Figuur 3.37
Een formulier met
DOM-elementen

13:31:30

Bestelformulier

Man: Vrouw:

vult uw voornaam in
vult uw achternaam in
vult uw leeftijd in
vult uw woonplaats in
vult uw postcode in
vult uw email in
vult uw 06 in
vult uw wachtwoord in
wachtwoord nogmaals

Bestelling

Smartphone:
Laptop:
Tablet:
Dag

Verzenden

Dit formulier maken we aan in de volgende opgave:

Opgave 99 Open **dom-events.html** en voeg het volgende formulier er aan toe:

```
<div style="width: 200px">
<form action="mailto:email@adres.nl"
      name="custform"
      method="post" >

    <fieldset style="padding-left:25px">
      <legend>Bestelformulier</legend>
      <label for="man">Man: </label>
      <input type="radio" name="geslacht" id="man" value="m"/>
      <label for="vrouw">Vrouw: </label>
      <input type="radio" name="geslacht" id="vrouw" value="v"/>

      <input required type="text" id="voornaam"
             placeholder="vult uw voornaam in" />
      <input required type="text" id="achternaam"
             placeholder="vult uw achternaam in" />
      <input required type="text" id="leeftijd"
             placeholder="vult uw leeftijd in" />
      <input required type="text" id="plaats"
             placeholder="vult uw woonplaats in" />
      <input required type="text" id="postcode"
             placeholder="vult uw postcode in" />
      <input required type="email" id="e-mail"
             placeholder="vult uw e-mail in" />
```

```
placeholder="vult uw email in" />
<input required type="text" id="06"
placeholder="vult uw 06 in" />
<input required type="password" id="password"
placeholder="vult uw wachtwoord in"/>
<input required type="password" id="password2"
placeholder="wachtwoord nogmaals" />
</fieldset>

<fieldset style="padding-left:25px">
<legend>Bestelling</legend>
<label for="smartphone"><br />Smartphone: </label>
<input type="checkbox" name="gadget"
id="smartphone" value="s" />
<label for="laptop"><br />Laptop: </label>
<input type="checkbox" name="gadget"
id="laptop" value="l" />
<label for="tablet"><br />Tablet: </label>
<input type="checkbox" name="gadget"
id="tablet" />
<br />
<select name="bezorging">
<option value="" default selected>
    Selecteer bezorging</option>
<option value="express">Express</option>
<option value="dag">Dag</option>
<option value="week">Week</option>
</select>
<br />
<div id="express-info" style="display:none">Express
bestelling</div>
<div id="dag-info" style="display:none">Dag bestelling</div>
<div id="week-info" style="display:none">Week bestelling</div>
<br />
<input type="submit" id="submit" name="submit"
value="Verzenden" />
</fieldset>
</form>
</div>
```

Je vangt form-events op in <form>-elementen. Bijvoorbeeld:

```
<form onsubmit="mijnfunctie()" name="custform"
method="POST">
of
<input type="text" onchange="mijnfunctie()"
name="leeftijd">
```

Opgave 100 Open **dom-events.html** en voeg de volgende `onchange`-event er aan toe.

```
<label for="man">Man: </label>
<input type="radio" name="geslacht" id="man" value="m"
       onchange=""
       if(document.custform.man.checked == true)
{
    alert('Welkom meneer!');
}

/>
<label for="vrouw">Vrouw: </label>
<input type="radio" name="geslacht" id="vrouw" value="v"
       onchange=""
       if(document.custform.vrouw.checked == true)
{
    alert('Welkom mevrouw!');
}
/>
```

In bovenstaande opgave hebben we een welkomstbericht gebaseerd op het geslacht van de user vertoond.

Opgave 101 Open **dom-events.html** en voeg de volgende `onchange`-event toe aan het element met `id="leeftijd"`:

```
<input required type="text" id="leeftijd"
       placeholder="vul uw leeftijd in"
       onchange=""
       if(document.custform.leeftijd.value <= 17)
{
    alert('Minderjarig');
}
/>
```

Hier controleren we of het veld `<input required type="text" id="leeftijd"` veranderd is. Als dat zo is, testen we of de nieuwe waarde gelijk aan of kleiner dan 17 is. Als dat het geval is maken we een alertbox met de tekst 'leerplichtig'.

Opgave 102 Open **dom-events.html** en voeg de volgende functionaliteit er aan toe:
Als de leeftijd boven de 55 is, dan verschijnt een alertbox met de melding:

Kijk naar onze seniorenprogramma's!

We zouden allerlei formuliercontroles kunnen uitvoeren binnen een formulier, maar dan eindigen we met een rommelige van HTML en JavaScript. Een betere oplossing is om je JavaScript-controles in een extern bestand te coderen.

In de volgende opgave zorgen we ervoor dat wanneer je op je verzendknop klikt, een JavaScript-formuliercontrole wordt uitgevoerd.

Opgave 103 Open **dom-events.html** en voeg de volgende form-events er aan toe:

```
<input type="submit" id="submit" name="submit"
value="Verzenden"
onmouseover="style.color='green'"
onmouseout="style.color='black'"
/>
```

Alle verplichte inputvelden en e-mailadressen worden gecontroleerd door HTML5. Maar we moeten zelf de volgende voorwaarden controleren:

1. De postcode mag geen spaties hebben;
2. De user moet 20 of ouder zijn;
3. Wanneer de user een product kiest, maken we een alertbox zoals: 'Tablet besteld'
4. Wanneer de user een bezorgmethode kiest, wordt een beschrijving van de keuze vertoond.

In de volgende opgave gaan we ervoor zorgen dat wanneer je op de verzendknop klikt, de functie `formcheck()` wordt uitgevoerd. Deze functie gaan we ook in de volgende opgaven coderen.

Opgave 104 Open **dom-events.html** verander het `<form>`-element als volgt:

```
<form action="mailto:email@adres.nl"
      name="custform"
      method="post"
      onsubmit="return(formcheck(this));"
      name="custform" method="post" >
```

Hier zeggen we dat de `onsubmit`-event de functie `formcheck()` moet uitvoeren. We zeggen ook dat als de returnwaarde van deze functie false is, het formulier niet wordt verstuurd. In de volgende opgave coderen we de functie `formcheck()`.

Opgave 105 Open **dom-events.js**. Codeer de functie `formcheck()`.

```
function formcheck(thisForm)
{
    postcode = thisForm.postcode.value;
    for(var i=0; i < postcode.length; i++)
    {
        var c = postcode.charAt(i);
        if(c == " ")
        {
            alert("Postcode mag geen spaties hebben");
            thisForm.postcode.focus();
            return false;
        }
    }
}
```

Deze functie controleert of de waarde van het inputveld postcode spaties heeft. Als dat zo is, wordt een alertbox vertoond en krijgt het inputveld postcode de focus, zodat de postcode opnieuw mag worden ingetypt.

Opgave 106 Open **dom-events.js** en voeg de volgende functionaliteit er aan toe:

Zorg ervoor dat de functie `formcheck()` controleert of de leeftijd onder de 20 jaar is en in dat geval de melding ‘Je moet 20 jaar of ouder zijn’ vertoont. De returnwaarde moet `false` zijn en je moet teruggaan naar het formulier met de focus op het leeftijdveld.

De status van een checkbox kunnen we ook controleren. Zo kunnen we afleiden welke opties de user heeft aangevinkt. In de volgende opgave kijken we naar drie elementen `<input type="checkbox" name="gadget">`. Het `<form>`-element heeft een array met de namen van alle formulier-elementen. De drie checkboxes kunnen we als volgt benaderen:

```
document.custform.gadget[0]
document.custform.gadget[1]
document.custform.gadget[2]
```

Opgave 107 Open **dom-events.html** en voeg de volgende functionaliteit er aan toe:

```
<input type="checkbox" name="gadget"
id="smartphone" value="s"
onchange="
if(document.custform.gadget[0].checked)
{
    alert('Smartphone besteld!');
}"
/>
```

Maak deze opgave af door alertboxes te maken voor ‘Laptop besteld’ en ‘Tablet besteld’.

Als laatste controleren we het menu met bezorgmethoden. In de volgende opgave controleren we of de bezorging ‘express’ geselecteerd is. Als dat zo is dan vertonen we meer informatie over deze keuze.

Opgave 108 Open **dom-events.html** en voeg de volgende functionaliteit er aan toe:

```
<select
onchange="
if(document.custform.bezorging.value == 'express')
{
    getElementById('express-info').style.display='block';
    getElementById('dag-info').style.display='none';
    getElementById('week-info').style.display='none';
}
"
name="bezorging" >
```

Maak deze opgave af door ervoor te zorgen dat indien gekozen, de elementen dag-info en week-info worden weergegeven.

Opgave 109 Open **dom-events.html** en zorg ervoor dat indien gekozen, de elementen dag-info en week-info worden weergegeven.

3.16.8 Keyboard-events

Keyboard-events zijn events die plaatsvinden wanneer je een toets op het toetsenbord indrukt. Hieronder volgt een tabel met de meeste voorkomende keyboard-events:

Events	Gebeurt wanneer
onkeydown	een toets wordt ingedrukt
onkeypress	een toets wordt ingedrukt en losgelaten
onkeyup	een toets wordt losgelaten

Je vangt een keyboard-event op tijdens het tikken. Bijvoorbeeld:

```
<form>
<input type="text" onkeyup="mijnfunctie()"
name="leeftijd">
```

In de volgende opgave vangen we een onkeyup-event op met een functie herhaling(). De functie zorgt ervoor dat als je je wachtwoord intikt, elke tik in het tweede wachtwoordveld wordt herhaald.

Opgave 110 Open **dom-events.html** en codeer een onkeyup-event als volgt:

```
<input required type="password" id="password"
placeholder="vult uw wachtwoord in"
onkeyup="herhalen()" />
```

Opgave 111 Open **dom-events.js** en codeer de functie herhalen() als volgt:

```
function herhalen(){
    document.custform.password2.value =
        document.custform.password.value;
}
```

3.16.9 Web-lab 22

Schrijf pseudocode voor het volgende algoritme (lingospel).

1. Kies een willekeurig woord;
bijvoorbeeld: algoritme
2. Toon twee willekeurige letters uit het woord en enkele X'en;
bijvoorbeeld: xxxoxxttxx
3. Vraag de speler om een gok;
bijvoorbeeld: automotor
4. Geef resultaat van de gok weer als volgt:
bijvoorbeeld:axxoxxxtxx (a,o en t zijn goed)
5. Herhaal stap 3 (maximaal 5 keer gokken).

Vertaal je pseudocode in JavaScript.

3.17 Cookies

Webpagina's zijn *stateless*. Dat betekent dat webpagina's geen geheugen hebben. Een webpagina weet dus niets van de vorige webpagina. Met andere woorden, de tweede, derde of honderdste keer dat je een webpagina bezoekt, is net alsof het de eerste keer is. Dit is een probleem als je informatie van de ene webpagina naar de andere wilt sturen. Om dit probleem op te lossen heeft Netscape cookies ontwikkeld. Met JavaScript kunnen we cookies maken, lezen en verwijderen.

3.17.1 Hoe werk je met cookies?

Een cookie is een ‘naam-waarde-construct’. Als je informatie tijdelijk wilt bewaren maak je een cookie. Een cookie moet een *naam* en een *waarde* hebben. Cookies worden op de harde schijf van de gebruiker opgeslagen als een tekstbestand. Een cookie maak je op de volgende manier:

```
document.cookie =
'naam=waarde; expires=Thu, 2 Aug 2014 20:47:11 UTC; path=/'
```

Je kunt de volgende stappen herkennen.

Eerst geef je een cookie een naam en waarde:

```
naam=waarde;
```

Vervolgens geef je de verloopdatum. Na deze datum wordt het cookie door de browser verwijderd:

```
expires= Thu, 2 Aug 2014 20:33:03 UTC;
```

Ten slotte geef je het pad naar je cookies: *path=/*. Deze property bepaalt de scope van de cookies. De scope bestaat uit de domeinnaam en de webpagina die de cookies aanmaakt. Bijvoorbeeld: *path=/voorbeeld.com/home* Een cookie voor de user maak je bijvoorbeeld als volgt:

```
document.cookie =
'user=admin; expires=Thu, 2 Aug 2014 20:47:11 UTC; path=/'
```

Een tweede cookie voor het wachtwoord maak je zo:

```
document.cookie =
'password=admin; expires=Thu, 2 Aug 2014 20:47:11 UTC; path=/'
```

Je mag de waarde van een cookie wijzigen door hetzelfde cookie opnieuw te maken, bijvoorbeeld:

```
document.cookie =
'password=9u23Qf;expires=Thu,2 Aug 2014 20:47:11 UTC;path=/'
```

Om een cookie te verwijderen, zet je de verloopdatum naar een datum eerder dan vandaag. De browser ziet dan dat het cookie is verlopen en vervolgens wordt het cookie verwijderd. Bijvoorbeeld:

```
document.cookie =
'password=9u23Qf;expires=Wed,1 Aug 2014 20:47:11 UTC;path=/'
```

3.17.2 JavaScript met cookies

Met JavaScript kunnen we cookies maken, lezen, wijzigen en verwijderen.

Deze cookies zijn dan beschikbaar voor de duur van je browsersessie. In de volgende opgave maken we een functie die nieuwe cookies aanmaakt.

Opgave 112 Codeer het volgende script en sla het op als **cookiefuncties.js**.

```
// function: maakCookie
// parameters: naam, waarde, dagen
// return :
// doel: een cookie maken
function maakCookie(naam, waarde, dagen)
{
    if (dagen)
    {
        var datum = new Date();
        datum.setTime(datum.getTime()+(dagen*24*60*60*1000));
        var verloopdatum = "; expires="+datum.toGMTString();
    }
    else
    {
        var verloopdatum = "";
    }
    document.cookie = naam+"="+waarde+verloopdatum+";path=/";
}
```

Om deze functie te gebruiken, moet je drie stukken informatie via de parameters doorgeven: de naam, de waarde en aantal dagen dat het cookie moet actief blijven. Bijvoorbeeld:

```
maakCookie("user","administrator",7);
```

Deze opdracht maakt het cookie **user** met de waarde **administrator** met een houdbaarheid van **zeven** dagen. In de volgende opgave maken we een inlogscript.

Opgave 113 Typ het volgende over en sla het op als **inloggen.html**.

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <meta http-equiv="Content-Type"
          content="text/html;
                  charset=UTF-8" />
    <title>Inlogscript</title>
    <script type="text/javascript" src="cookiefuncties.js">
    </script>
    <style type="text/css">
        div{
            height:120px;
            width:200px;
            padding: 20px;
            border-width: 1px;
```

```

        border-style: solid;
        border-color:gray;
        background-color: rgb(217,217,0);
    }
</style>
</head>
<body>
    <div>
        <form onsubmit=
            "maakCookie('user',document.inlog.gebruiker.value,1);
            maakCookie('password',document.inlog.wachtwoord.value,1);"
            name="inlog" action="welkom.html" method="post" >
            Email-adres:
            <input type="email" size="20" id="gebruiker" />
            Wachtwoord:
            <input type="password" size="20" id="wachtwoord" />
            <hr />
            <input type="submit" id="submit"
                   name="submit" value="Verzenden" />
        </form>
    </div>
</body>
</html>

```

In de vorige opgave hebben we een inlogscript gecodeerd. Op het moment van de `onsubmit`-event maken we het user-cookie en het password-cookie aan.

Figuur 3.38
Inlogformulier

The image shows a simple HTML form enclosed in a light gray box. It has two text input fields. The first field is labeled 'Email-adres:' and contains the text 'aqa@nmn.nm'. The second field is labeled 'Wachtwoord:' and contains five black dots ('....'). Below these fields is a single button labeled 'Verzenden'.

Dit inlogscript wordt daarna gelinkt met de volgende pagina welkom.html:

Opgave 114 Open een nieuw bestand en sla het op als **welkom.html**. Codeer de volgende welkompagina:

```

<!DOCTYPE html>
<html lang="nl">
<head>
    <meta http-equiv="Content-Type"
          content="text/html;
          charset=UTF-8" />
    <title>Welkom</title>
    <script type="text/javascript" src="cookiefuncties.js">
    </script>
</head>
<body>
    <script type="text/javascript">

```

```
        var newuser = leesCookie("user");
        document.writeln("Welkom " + newuser +
        " U bent ingelogd!!!!");
    </script>
</body>
</html>
```

In deze pagina willen we in `cookiefuncties.js` de functie `leesCookie()` uitvoeren. Deze functie moet de naam van de gebruiker vanuit het user-cookie lezen om de gebruiker te verwelkomen. De functie `leesCookie()` coderen we in de volgende opgave:

Opgave 115 Open `cookiefuncties.js` en voeg de volgende functie er aan toe:

```
// function: leesCookie
// parameters: naam
// return : waarde van de cookie
// doel: de waarde van een cookie retourneren
function leesCookie(naam)
{
    var naamCookie = naam + "=";
    var cookieArray = document.cookie.split(';");
    for(var i=0; i < cookieArray.length; i++)
    {
        var dezeCookie = cookieArray[i];
        while (dezeCookie.charAt(0)==' ')
        {
            dezeCookie = dezeCookie.
                substring(1,dezeCookie.length);
        }
        if (dezeCookie.indexOf(naamCookie) == 0)
        {
            return dezeCookie.substring(naamCookie.
                length,dezeCookie.length);
        }
    }
    return null;
}
```

Om deze functie te gebruiken, moet je de naam van een cookie via de input-parameter geven. Bijvoorbeeld:

```
gebruikersnaam = leesCookie("user");
```

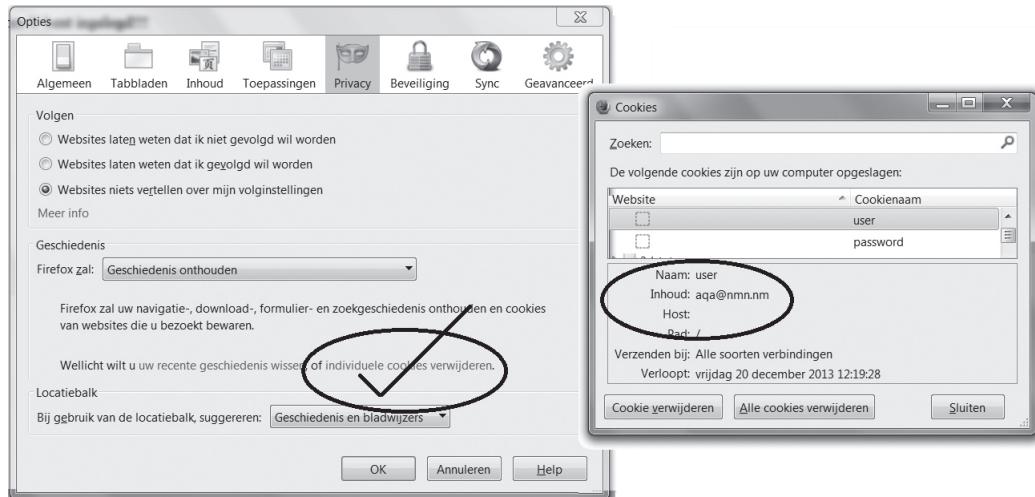
zal de waarde van de cookie **user** retourneren. Met de functie `split()` maak je een array met daarin alle cookies. Met de for-lus zoek je alle cookies in je array. Met de while-lus en de functie `substring()` verwijder je alle spaties vóór de cookienaam. Met de if-opdracht en de functie `indexOf()` controleer je of de naam van het cookie gelijk is aan “user”. Met `return` geeft je de waarde van het cookie terug.

Als alles goed verloopt, zul je het volgende resultaat zien:

Welkom aqa@nmn.nm U bent ingelogd!!!

3.17.3 Browser en cookies

Als je nu op het menu *Opties* van je browser klikt, komt het volgende venster te voorschijn.



Figuur 3.39 Cookies tonen

Controleer in je eigen browser of de cookie 'user' inderdaad is aangemaakt.

Klik op *Privacy* en op de link *individuele cookies verwijderen* en als het goed is zie je hier de twee cookies 'user' en 'password' die je net gemaakt hebt. In de volgende opgave maken we een functie om cookies te verwijderen.

Opgave 116 Open **cookiefuncties.js** en voeg de volgende functie er aan toe:

```
// function: verwijderCookie
// parameters: naam
// return :
// doel: cookie verwijderen

function verwijderCookie(naam) {
    maakCookie(naam, "", -1);
}
```

Met de functie `verwijderCookie()` zet je de verloopdatum naar een dag eerder. Als de browser ziet dat de datum verlopen is, wordt het cookie verwijderd.

Opgave 117 Maak een `<a>`-link vanuit de pagina `welkom.html` naar een pagina **uitloggen.html**. Maak de pagina `uitloggen.html`. In deze pagina voer je de functie `verwijderCookie()` uit om de cookies 'user' en 'password' te verwijderen. Toon de volgende tekst: 'U bent uitgelogd en cookies zijn verwijderd.'

3.18 Een winkelwagentje in JavaScript

In dit project maken we een winkelwagentje, een client-side winkelwagenapplicatie met JavaScript voor een webwinkel. Begin met het coderen van de volgende scripts. Gebruik je eigen afbeeldingen en knoppen. Dit eerste script sla je op als **xxlcomputers.html**:

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <meta http-equiv="Content-Type"
          content="text/html;
          charset=UTF-8" />
    <title>XXL-Computers</title>
    <script type="text/javascript" src="cookiefuncties.js">
    </script>
    <script type="text/javascript" src ="shoppingcart.js">
    </script>
</head>
<body>
<table name="toshiba" border=0 cellpadding=0
cellspacing=0 width=100%
<tr><td></td></tr>
<tr><td>Toshiba Satellite A100 Prijs 999.99</td></tr>
<tr><td><!--Begin Toshiba gegevens-->
    <form name="orderform" action="managecart.html"
        method="get" onsubmit="bestellen(this);">
        Aantal: <input type="text" size=2 maxlength=3
        id="aantal" value="1" />
        <input type="image" src="inwagentje.bmp" />
        <input type="hidden" id="id" value="001" />
        <input type="hidden" id="merk" value="Toshiba" />
        <input type="hidden" id="model" value="Satellite" />
        <input type="hidden" id="prijs" value="999.99" />
    </form>
</td><!--Einde Toshiba --></tr>
</table>
<hr />

<table name="acer" border=0 cellpadding=0
cellspacing=0 width=100%
<tr><td></td></tr>
<tr><td>Acer Aspire 3072 MB - 160 GB Prijs 529.00</td></tr>
<tr><td><!--Begin Aspire gegevens-->
    <form name="orderform" action="managecart.html"
        method="get" onsubmit="bestellen(this);">
        Aantal: <input type="text" size=2 maxlength=3
        id="aantal" value="1" />
        <input type="image" src="inwagentje.bmp" />
        <input type="hidden" id="id" value="002" />
        <input type="hidden" id="merk" value="Acer" />
        <input type="hidden" id="model" value="Aspire" />
        <input type="hidden" id="prijs" value="529.99" />
    </form>
```

```
</td><!--Einde Aspire --></tr>
</table>
<hr />
</body>
</html>
```

In dit script maken we twee tabellen met in elke tabel een bestelformulier. Van dit formulier zijn maar een paar elementen zichtbaar. De andere elementen zijn van het type="hidden". Na klikken op de bestelknop worden de gegevens aan het winkelwagentje toegevoegd met onsubmit="bestellen(this)". Daarna wordt het winkelwagentje weer gegeven met action="manageCart.html". Het resultaat van de pagina xxlcomputers.html moet er zo uitzien:

Figuur 3.40

Het winkelwagentje van xxlcomputers.html

The screenshot shows a web page with two product entries in a shopping cart. Each entry includes a small image of the laptop, the model name, the original price (struck through), the discounted price, and a quantity input field set to 1. A button labeled "in winkelwagentje" is present for each item.

Product	Oorspronkelijke Prijs	Gebruikte Prijs	Aantal	Toevoegen aan winkelwagentje
Toshiba Satellite A100	999.99	529,-	1	<input type="button" value="in winkelwagentje"/>
Acer Aspire 3072 MB - 160 GB	599,-	529,-	1	<input type="button" value="in winkelwagentje"/>

Nu gaan we wat functies toevoegen aan het winkelwagentje. Typ het volgende over en sla het op als **shoppingcart.js**:

```
// FUNCTION: bestellen
// PARAMETERS: Form Object
// RETURNS: Cookie met bestelling info...
// PURPOSE: Voeg bestelling naar shopping cart toe
function bestellen(thisForm)
{
    intBesteldeItems = leesCookie("besteldeItems");
    if(intBesteldeItems == null) {intBesteldeItems = 0;}
    intBesteldeItems++;
    if ( intBesteldeItems > 5 )
    {
        alert("Shoppingcart is vol, Ga naar checkout.");
    }
    else
    {
        productInfo = thisForm.id.value + "|"
        + thisForm.merk.value + "|"
        + thisForm.model.value + "|"
        + thisForm.prijs.value + "|"
        + thisForm.aantal.value;
    }
}
```

```
item = "item" + intBesteldeItems;
maakCookie (item, productInfo,1);
maakCookie ("besteldeItems", intBesteldeItems,1);
notice = thisForm.aantal.value + " "
+ thisForm.merk.value
+ " in winkelwagen.";
alert(notice);
}
}
```

In dit script maakt de functie `bestellen()` een cookie met de naam `item` en een cookie met de naam `besteldeItems` met een aantal bestelde producten:

```
maakCookie (item, productInfo,1);
maakCookie ("besteldeItems", intBesteldeItems,1);
```

Klik je bijvoorbeeld op de winkelwagentjesknop, dan wordt een cookie gemaakt met de naam `item1` en met de gegevens van de laptop. De volgende keer dat je erop klikt wordt het cookie `item2` gemaakt, enzovoort. De naam van het tweede cookie `besteldeItems` houdt het aantal bestelde items bij. Telkens als je op de bestelknop klikt, wordt `besteldeItems` erbij opgeteld.

Open **shoppingcart.js** en voeg de volgende functie eraan toe:

```
// FUNCTION: cartWeergeven
// PARAMETERS: Null
// RETURNS: Productentabel
// PURPOSE: Geeft productentabel weer in html
// formaat
function cartWeergeven()
{
    intBesteldeItems = 0;
    intBesteldeItems = leesCookie("besteldeItems");
    tabelrij = "";
    for (i = 1; i <= intBesteldeItems; i++)
    {
        item = "item" + i;
        dezeCookie = "";
        dezeCookie = leesCookie(item);
        velden = new Array();
        velden = dezeCookie.split("|");
        tabelrij += "<tr>" +
                    "<td>" + velden[0] + "</td>" +
                    "<td>" + velden[1] + "</td>" +
                    "<td>" + velden[2] + "</td>" +
                    "<td>" + velden[3] + "</td>" +
                    "<td>" + velden[4] + "</td>" +
                    "</tr>";
    }
    document.write(tabelrij);
}
```

In bovenstaand script gebruikt de functie `cartWeergeven()` een for-lus om alle cookies te lezen en in tabelvorm weer te geven.

Typ het volgende over en sla het op als **managecart.html**:

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <meta http-equiv="Content-Type"
          content="text/html;
          charset=UTF-8" />
    <title>Manage Cart</title>
    <script type="text/javascript" src="cookiefuncties.js">
    </script>
    <script type="text/javascript" src ="shoppingcart.js">
    </script>
</head>
<body>
<p>Overzicht van Uw shoppingcart:</p>
<p>
<form action="mailto:webshop@xxlcomputers.nl"
      enctype="text/plain" /*onSubmit="return checkForm(this);"/*
      method="post">
      <table border=1>
      <tr>
          <td bgcolor="#cccccc"><b> Product ID </b></td>
          <td bgcolor="#cccccc"><b> Merk </b></td>
          <td bgcolor="#cccccc"><b> Model </b></td>
          <td bgcolor="#cccccc"><b> Prijs </b></td>
          <td bgcolor="#cccccc"><b> Aantal</b></td>
      </tr>
      <script>
          cartWeergeven();
          bestellingsFormInvullen();
      </script>
      </table>
      <br /><br />
      // klantgegevens
      <table border=1 width="500">
      <tr>
          <td>Naam: <input required type="text" size="18"
              name="naam" /> </td>
      </tr>
      <tr>
          <td>Achternaam: <input required type="text" size="18"
              name="achternaam"></td>
      </tr>
      <tr>
          <td>E-mail: <input required type="email" size="49"
              name="email"></td>
      </tr>
      </table>
      <a href="xxl computers.html"> </a>
```

```

<a href="xxl computers.html"> </a>
<input type="image" src="checkout.jpg" value="send" />
</form>
</body>
</html>

```

Het resultaat van het bestand managecart.html moet er zo uitzien:

Figuur 3.41

Het resultaat van
managecart.html

Overzicht van Uw shoppingcart:

Product ID	Merk	Model	Prijs	Aantal
001	Toshiba	Satellite	999.99	11
002	Acer	Aspire	529.99	2

Naam:	<input type="text"/>
Achternaam:	<input type="text"/>
e-mail:	<input type="text"/>
<input type="button" value="Terug..."/> <input type="button" value="Reset"/> <input type="button" value="Checkout"/>	

Klanten willen hun bestelling ook wel eens aanpassen of verwijderen. We gaan nog een functie toevoegen.

Open nu **cookiefuncties.js** en voeg de volgende functie er aan toe:

```

// FUNCTION: verwijderAlleCookies
// PARAMETERS: Null
// RETURNS: Null
// PURPOSE: Alle cookies verwijderen
function verwijderAlleCookies() {
    besteldeItems = leesCookie("besteldeItems");
    for (i = 1; i <= besteldeItems; i++)
    {
        item = "item" + i;
        verwijderCookie(item);
    }
    verwijderCookie("besteldeItems");
}

```

Bovenstaande functie `verwijderAlleCookies()` gebruikt een for-lus om alle `item`-cookies te verwijderen. De functie wordt uitgevoerd wanneer op de knop Reset wordt geklikt.

Er moeten natuurlijk ook bestelgegevens worden ingevuld. Open **shoppingcart.js** en voeg de volgende functie er aan toe:

```

// FUNCTION: bestellingsFormInvullen
// PARAMETERS: Null
// RETURNS: ingevuld formulier
// PURPOSE: bestellingsformulier invullen
function bestellingsFormInvullen() {
    intBesteldeItems = 0;
    intBesteldeItems = leesCookie("besteldeItems");
    for (i = 1; i <= intBesteldeItems; i++)

```

```

{
    item = "item" + i;
    dezeCookie = "";
    dezeCookie = leesCookie(item);
    velden = new Array();
    velden = dezeCookie.split("|");
    document.write(
        "<input type=hidden name=\"ID_" + i +
        "\" value=\"" + velden[0] + "\">");
    document.write(
        "<input type=hidden name=\"MERK_" + i +
        "\" value=\"" + velden[1] + "\">");
    document.write("<input type=hidden name=\"MODEL_" + i +
        "\" value=\"" + velden[2] + "\">");
    document.write("<input type=hidden name=\"PRIJS_" + i +
        "\" value=\"" + velden[3] + "\">");
    document.write("<input type=hidden name=\"AANTAL_" + i +
        "\" value=\"" + velden[4] + "\">");
}
}

```

Bovenstaande functie bestellingsformInvullen() leest alle cookies en vult de gegevens in, in een bestelformulier. De functie wordt uitgevoerd na elk besteld item.

3.18.1 Web-lab 23

Om deze applicatie af te ronden maak je het volgende:

- Voeg drie nieuwe laptops toe.
- Voeg een nieuw gegeven Operating System toe voor alle producten, bijvoorbeeld: Windows, Mac OS X of Linux.
- Voeg een nieuwe kolom *Bedrag (prijs * aantal)* toe aan de functie cartWeergeven().
- Voeg een nieuwe kolom *Del* toe aan de functie cartWeergeven().

Plaats hier een afbeelding. Koppel hieraan de functie verwijderCookie(), zodat als op de afbeelding wordt geklikt, de desbetreffende laptop verwijderd wordt uit het winkelwagentje.

Het resultaat moet er als volgt uitzien:

Figuur 3.42

Het resultaat van
web-lab 23

Overzicht van Uw shoppingcart:							
Product ID	Merk	Model	OS	Aantal	Prijs	Bedrag	Delete
001	Toshiba	Satellite	Linux	999.99	2	1999.98	
002	Acer	Aspire	Windows	529.99	10	5299.9	

Naam: <input type="text"/>
Achternaam: <input type="text"/>
E-mail: <input type="text"/>
<input type="button" value="Terug..."/> <input type="button" value="Reset"/> <input type="button" value="Checkout"/>

4.1 Inleiding PHP

Voorkennis

Kennis van en ervaring met webapplicatieontwikkeling, met name HTML, CSS en JavaScript is vereist.

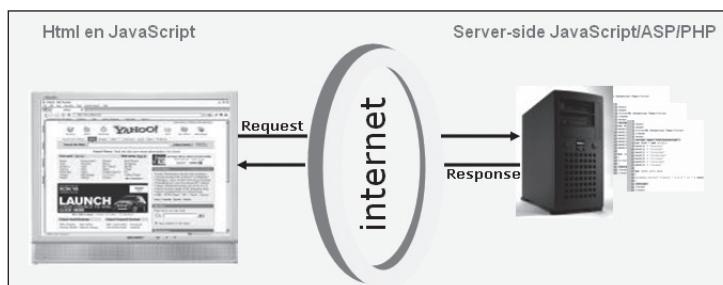
Het client-servermodel is een model waar twee computers in samenwerking twee of meer programma's uitvoeren, bijvoorbeeld, e-mail of internetbankieren.

Dit model werkt als volgt: een clientprogramma doet een aanvraag (request) bij het serverprogramma. Het serverprogramma voert hierop handelingen uit en geeft informatie terug aan de client. Bijvoorbeeld, een webserver ontvangt requests van verschillende webbrowsers. De server voert acties uit en geeft de informatie terug in de vorm van webpagina's. Deze webserverprogrammatuur noemen we een webapplicatie. In de rest van dit boek houden we ons bezig met het opbouwen van webapplicaties met PHP.

4.1.1 Wat is PHP?

PHP (oorspronkelijk *Personal Home Page*) is een server-side scriptingtaal speciaal ontworpen voor het web. Je kunt PHP-code binnen een html-pagina embedden. Deze code wordt uitgevoerd door de webserver wanneer iemand jouw webpagina bezoekt. PHP is ontworpen door Rasmus Lerdorf in 1994. Het is een open source-product en gratis te downloaden. De homepage voor PHP vind je op: <http://www.php.net>.

Figuur 4.1
Webapplicatie

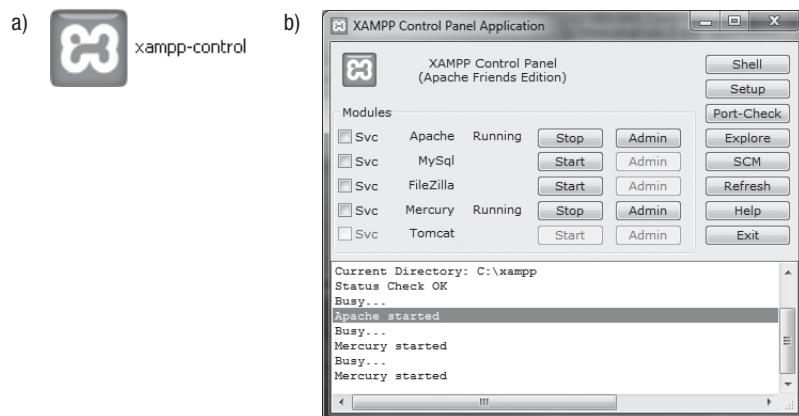


4.1.2 PHP installeren

Een webserver is de software die webpagina's aanbiedt ('serveert') aan de browsers van de gebruikers. PHP draait op de Apache-webserver. Om PHP-scripts te kunnen draaien moet je eerst de Apache-webserver op je computer installeren. De meest populaire open-source webserver is de XAMPP-server. XAMPP bestaat voor Multiplatform Apache MySQL PHP en Pearl. Je kunt XAMPP op een webhost installeren. Maar voor het ontwikkelen en testen van je websites kun je XAMPP op je eigen pc of laptop installeren. Je kunt de server downloaden vanaf de site www.apachefriends.org

Download en installeer XAMPP. Na het installeren start je XAMPP op door op het icon van XAMPP-control te klikken (zie figuur 4.2a), waarna de XAMPP Control Panel Application verschijnt (figuur 4.2b).

Figuur 4.2
De XAMPP Control Panel Application



In het Controlpaneel kun de Apache en MySQL services zien. Om de Apache server te starten klik je op Start en Admin. Om te testen of de installatie goed verlopen is typ je in het adresvakje van de browser het volgende:

`http://localhost`

4.1.3 Eerste PHP-script

De webserver herkent een PHP-script tussen de volgende tags:

Syntaxis:

```
<?php  
?  
?
```

Voorbeeld 4.1

Zorg ervoor dat **les1.php** er als volgt uitziet:

```
<!DOCTYPE html>  
<html lang="nl">  
<head>
```

```
<title>Mijn php-script</title>
</head>
<body>
<?php
phpinfo();
?>
</body>
</html >
```

Maak een nieuwe map **unit4** in de map **htdocs** van XAMPP en sla **les1.php** in deze map op. Om je eerste PHP-script te kunnen testen, typ je het volgende in de adresbalk van je browser:

<http://localhost/unit4/les1.php>

Figuur 4.3

Het resultaat van les 1. php moet er als volgt uitzien

System	Windows NT JESSIKA 5.1 build 2600
Build Date	Aug 24 2003 22:01:16
Server API	Apache
Virtual Directory Support	enabled
Configuration File (php.ini) Path	c:\program files\easyphp1-7\apache\php.ini
PHP API	20020918
PHP Extension	20020429
Zend Extension	20021010
Debug Build	no
Thread Safety	enabled
Registered PHP Streams	php, http, ftp, compress.zlib

- De `phpinfo()`-functie is een standaard-PHP-functie die de webpagina uit figuur 4.3 weergeeft.
- Alle instructies binnen de `<?php ?>`-tags eindigen met een puntkomma (`;`).
- Een PHP-script moet in de `<body>`-sectie van je html-document ingebed worden.

4.2 De PHP-programmeertaal

4.2.1 Taalcomponenten

De PHP-syntaxis is hetzelfde als de syntaxis van andere programmeertalen, zoals C en Java. De volgende tabel is een samenvatting van de PHP-syntaxis:

Taalcomponent	PHP-syntaxis
Open en sluit script tags	<?php ?>
Blok	of <? ?>
Commentaar	{ }
Declaratie van een variabele	//mijn commentaar
Strings	var \$mijnvar = 0; of \$mijnvar = 0; var \$mijntekst = " tekst tussen aanhalingstekens";

4.2.2 De echo-opdracht

Om output weer te geven gebruik je de `echo`-opdracht. De syntaxis is:

```
echo("jouw tekst");
```

Voorbeeld 4.2

Zorg ervoor dat **voorbeeld.php** er als volgt uitziet:

```
<!DOCTYPE html>
<html lang="nl">
<head>
<title>Mijn php-script</title>
</head>
<body>
<?php
echo("deze tekst weergeven");
?>
</body>
</html >
```

Figuur 4.4

Het resultaat van voorbeeld 4.2



- Opgave** 1 Maak een nieuw script (**les2.php**) en `echo()` de volgende tekst: **met deze tekst is les 2 afgerond.**

Figuur 4.5
Het resultaat van opgave 1



4.3 PHP-variabelen

4.3.1 Wat is een variabele?

Een PHP-variabele is een data container met een naam en een waarde.

De naam van de variabele is het adres van de variabele in het computergeheugen. Op dit adres kun je data tijdelijk bewaren. Een variabele kun je voorstellen als een laatje met een naam in het geheugen van de computer. Door middel van de naam kun je de inhoud van dat laatje bekijken of wijzigen.

4.3.2 Namen van variabelen

De naam van een variabele:

- begint altijd met het \$-teken;
- mag letters, getallen of underscores hebben;
- is hoofdlettergevoelig, bijvoorbeeld `$adres` en `$Adres` zijn twee verschillende variabelen.

In de volgende tabel declareer je drie variabelen:

variabeledeclaratie	resultaat in het geheugen
<code>var \$naam="Carl";</code>	<div style="display: flex; align-items: center;"> \$naam → Carl </div>
<code>var \$leeftijd=16;</code>	<div style="display: flex; align-items: center;"> \$leeftijd → 16 </div>
<code>var \$leerplichtig=true;</code>	<div style="display: flex; align-items: center;"> \$leerplichtig → true </div>

De waarde van een variabele kun je wijzigen. Bijvoorbeeld:

<code>\$leeftijd = leeftijd + 2;</code>	<div style="display: flex; align-items: center;"> \$leeftijd → 18 </div>
---	---

Een variabele mag de volgende datatypen en waarden hebben:

datatype	betekenis	voorbeeld
string	tekst	" Carl"
boolean	waar/niet waar	true/false
integer	geheel getal	18
float	drijvendecomma	1.79
array	verzameling	[2,4,6,8]

Hieronder zie je de belangrijkste rekenkundige operatoren:

operator	betekenis	voorbeeld
++	plus 1	teller++;
--	min 1	teller--;
+	optellen	\$totaal=\$a+\$b;
-	aftrekken	\$teller=\$teller-3;
*	vermenigvuldigen	\$vlakte=\$L*\$B;
/	delen	\$totaal=\$a/\$b;
.	tekst samenvoegen	\$msg = "Error" . "001";

Voorbeeld 4.3

Zorg ervoor dat **voorbeeld.php** er als volgt uitziet:

```
<!DOCTYPE html>
<html lang="nl">
<head>
<title>Mijn php-script</title>
</head>
<body>
<h3>Variabelen</h3>
<?php
$breedte = 10;
$lengte = 11;
$hoogte = 5;
$oppervlakte = $lengte * $breedte;
$volume = $oppervlakte * $hoogte;
echo("Oppervlakte is: " . $oppervlakte . "<br />");
echo("Volume is: " . $volume . "<br />");
echo("Half volume is: " . $volume / 2 . "<br />");
?>
</body>
</html >
```

Figuur 4.6

Het resultaat van voorbeeld 4.3

**Opgave**

- 2** Verander bovenstaand **voorbeeld.php** in **les3.php**, zodat:
\$lengte = 2*\$breedte

Figuur 4.7

Het resultaat van opgave 2



4.3.3 Datatypes

PHP werkt, zoals iedere programmeertaal, met de volgende datatypes:

- **String** voor teksten of tekenreeksen;
- **Integer** voor gehele getallen, bijvoorbeeld 3 of 500;
- **Float** voor reële getallen bijvoorbeeld 67 of 123.465;
- **Boolean** voor **true** of **false** waarden;
- **NULL** voor variabelen die (nog) geen waarde hebben.

Het is belangrijk om goed te begrijpen waanneer en waarom je een specifiek datatype moet gebruiken. Hier volgt een overzicht van datatypen.

Het datatype string

Strings zijn tekenreeksen. Een string geef je aan als een tekst tussen aanhalingstekens, bijvoorbeeld "tekst". De meeste informatie in databases of webapplicaties bestaat in de vorm van teksten. Bijvoorbeeld NAW-gegevens, productinformatie, enz.

Typ het volgende over en sla het op als **voorbeeld.php**.

```
<?
$naam = "Carl";
$adres = "Kruislaan 111";
$woonplaats = "Utrecht";
?>
```

Samenvoegen van twee tekstelementen

Met de operator punt (.) kun je twee tekst elementen samenvoegen. Bijvoorbeeld: "boek" . "title" geeft als resultaat de string "boek title". Open **voorbeeld.php** en voeg de volgende codes eraan toe:

```
<?
$naam = "Carl";
$adres = " Kruislaan 111";
$woonplaats = " Utrecht";
$naw = $naam . $adres . $woonplaats;
echo("Gegevens: $naw");
?>
```

Je kunt binnen een string ook variabelen plaatsen, bijvoorbeeld:

```
echo("Gegevens: $naw");
```

De waarde van de variabele wordt als eerste geëvalueerd en daarna de string.

De heredoc syntaxis (<<<)

Met de heredoc syntaxis kun je lange teksten maken. Je hoeft alleen een eindmarkering te specificeren en aan het einde van je tekst de eindmarkering te plaatsen. Open **voorbeeld.php** en voeg de volgende codes eraan toe:

```
<?
$naam = "Carl";
$adres = " Kruislaan 111";
$woonplaats = " Utrecht";
$naw = $naam . $adres . $woonplaats;
//resultaat "Carl Kruislaan 111 Utrecht"
echo("Gegevens: $naw");
echo <<<EIND
Salaris specificatie<br />maand: november jaar:2010
plaats:$woonplaats<br />
Algemene gegevens:
EIND;
?>
```

Het datatype integer

Een integer is een getal zonder decimale komma (of decimale punt), bijvoorbeeld het getal 6. Een getal met het datatype integer kan positief of negatief zijn. Open **voorbeeld.php** en voeg de volgende codes eraan toe:

```
<?
$naam = "Carl";
$adres = " Kruislaan 111";
$woonplaats = " Utrecht";
$naw = $naam . $adres . $woonplaats;
//resultaat "Carl Kruislaan 111 Utrecht"
echo("Gegevens: $naw");
echo <<<EIND
Salaris specificatie<br />maand: november jaar:2010
plaats:$woonplaats<br />
Algemene gegevens:
EIND;

$werkdagen = 5;
echo("<br />werkdagen:" . $werkdagen);
//resultaat werkdagen:5
?>
```

Het datatype float

Een getal van het datatype float is een getal met een decimale punt. Open **voorbeeld.php** en voeg de volgende codes eraan toe:

```
<?
$naam = "Carl";
$adres = " Kruislaan 111";
$woonplaats = " Utrecht";
$naw = $naam . $adres . $woonplaats;
//resultaat "Carl Kruislaan 111 Utrecht"
echo("Gegevens: $naw");
echo <<<EIND
Salaris specificatie<br />maand: november jaar:2010
plaats:$woonplaats<br />
Algemene gegevens:
EIND;

$werkdagen = 5;
echo("<br />werkdagen:" . $werkdagen);
//resultaat werkdagen:5

$uurtarief = 13.432;
echo ("<br />uurtarief is:" . $uurtarief);
//resultaat uurtarief is:13.432
?>
```

Getallen van het datatype float kunnen geformatteerd worden voor het printen, bijvoorbeeld met drie of twee decimale posities.

De functie printf()

De functie `printf()` formateert de waarde van een variabele in een gegeven print formaat. Open **voorbeeld.php** en voeg de volgende codes eraan toe:

```
<?
$uurtarief = 13.432;
echo ("<br />uurtarief is:" . $uurtarief);
//resultaat uurtarief is:13.432

printf("<br />uurtarief is: %.2f", $uurtarief);
//resultaat uurtarief is:13.43
?>
```

De functie `printf("uurtarief is: %.2f", $uurtarief)` formateert de float-waarde van de variabele `$uurtarief` met het formaat `%.2f`, ofwel met twee decimale posities.

De conversie van float naar integer met intval()

Met de functie `intval()` maken we conversies van float datatypes naar integer datatypes. Open **voorbeeld.php** en voeg de volgende codes eraan toe:

```
<?
printf("<br />uurtarief is: %.2f", $uurtarief);
//resultaat uurtarief is:13.43
```

```
$uren = 45.6;
echo ("<br />aantal uren is:" . intval($uren));
//resultaat aantal uren is:45
?>
```

De conversie van integer naar float met floatval()

Met de functie `floatval()` maken we conversies van het datatype integer naar float. Open [voorbeeld.php](#) en voeg de volgende codes eraan toe:

```
<?
$salaris = uren * uurtarief;
printf("salaris is: %.2f", $salaris);
//resultaat salaris is:612.50

printf("<br />aantal werkdagen is: %.1f",
floatval($werkdagen));
//resultaat aantal werkdagen is:5.0
?>
```

Het datatype boolean

Het datatype boolean mag alleen twee waarden krijgen: true of false (waar of onwaar) Op basis van de waarde van boolean variabelen kunnen we straks interessante controles programmeren.

Open [voorbeeld.php](#) en voeg de volgende codes eraan toe:

```
<?

printf("<br />aantal werkdagen is: %.1f",
floatval($werkdagen));
//resultaat aantal werkdagen is:5.0

$singel = true;
$sportief = true;
echo ("<br />singel:$singel sportief:$sportief ");
//resultaat singel:1 sportief:1
?>
```

De waarde true wordt als het getal 1 weergegeven. De waarde false wordt als 0 weergegeven.

PHP code camp - Lab 1

Datatypes

In deze lab-opdracht maak je variabelen voor het tijdelijk bewaren van gegevens. Kijk naar de volgende tabel met informatie over laptops.

Merk	Model	Operating system	Voorraad	Prijs
Toshiba Satellite	A100	Windows XP	80	999
Acer Aspire	5732Z	Linux	0	888

Deze gegevens kunnen we in variabelen tijdelijke bewaren.

Stap 1: Typ het volgen php-script over en sla het op als **lab01.php**.

```
<h3>php lab 01</h3>
<?php
$merk = " Toshiba Satellite ";
$model = " A100 ";
$os = " Windows XP ";
$voorraad = 80;
$prijs = 999;
$totaal = 0;
$totaal += $prijs;

echo("<table border='1'> ".
"<caption>
<strong>SML laptops</strong>
</caption>
<thead>
<tr><th>Merk</th><th>Model</th><th>Operating system
</th><th>Voorraad</th><th>Prijs</th></tr>
</thead>
<tbody>
<tr>
<td>" . $merk . "</td>" .
"<td>" . $model . "</td>" .
"<td>" . $os . "</td>" .
"<td>" . $voorraad . "</td>" .
"<td>" . $prijs . "</td>" .
"</tr>
<tfoot>
<tr><td colspan='4'>Totaal</td><td>" . $totaal . "
</td></tr></tfoot></table>");
?>
```

Stap 2: Het resultaat van bovenstaand script moet er als volgt uitzien:

Figuur 4.8

Het resultaat voor
één laptop

Merk	Model	Operating system	Voorraad	Prijs
Toshiba Satellite	A100	Windows XP	80	999
Totaal				999

Stap 3: Maak variabelen voor de Acer Aspire laptop en geef deze gevenges weer zoals hieronder.

Stap 4: Reken het totale bedrag van de twee laptops uit. Het resultaat moet er als volgt uitzien:

Figuur 4.9

Het resultaat voor twee laptops

The screenshot shows a Mozilla Firefox window with the URL `http://127.0.0.1:4001/phplabs/lab01.php`. The page title is "php lab 01". Below it is a table titled "SML laptops" with the following data:

Merk	Model	Operating system	Voorraad	Prijs
Toshiba Satellite	A100	Windows XP	80	999
Acer Aspire	5732Z	Linux	0	888
			Totaal	1887

At the bottom of the browser window, there is a status bar with the word "Klaar".

4.4 Arrays

Naast variabelen heb je in PHP ook datastructuren nodig waar je informatie tijdelijk kunt bewaren. Een van de eenvoudigste datastructuren is de array.

4.4.1 Wat is een array?

Een array is een serie van dezelfde soort dataelementen. Ieder element heeft een positie in de array. De positie van een element in de array geef je aan met een index. Een array kun je je voorstellen als een ladekastje met een naam in het geheugen van de computer. Elk laatje heeft een eigen volgnummer. Door middel van de naam en het volgnummer (index) kun je de inhoud van een laatje bekijken of wijzigen.

4.4.2 Array declareren

De PHP-syntaxis voor het declareren van een array is:

```
$arraynaam=array("waarde1","waarde2","waarde3",...);
```

Of

```
$arraynaam[0] = "waarde1";
$arraynaam[1] = "waarde2";
$arraynaam[2] = "waarde3";
```

Je zou een array als volgt in het computergeheugen kunnen zien:

\$arraynaam		
[0]	[1]	[2]
waarde1	waarde2	waarde3

Bijvoorbeeld:

```
$ingrediënt = array("melk", "olie", "suiker");
```

Of

```
$ingrediënt[0] = "melk";
$ingrediënt[1] = "olie";
$ingrediënt[2] = "suiker";
```

Deze instructies genereren de volgende array:

\$ingrediënt		
[0]	[1]	[2]
melk	olie	suiker

Om toegang te krijgen tot een element van een array gebruik je de array-naam plus de [elementindex]. In PHP begin je altijd met index [0].

Bijvoorbeeld de volgende echo-opdracht zal **suiker** weergeven:

```
echo($ingrediënt[2]);
```

De waarde van een array-element wijzig je met de operator =.

Bijvoorbeeld: de volgende instructie verandert het tweede element van **olie** in **koffie**:

```
$ingrediënt[1] = "koffie";
```

Voorbeeld 4.4

Zorg ervoor dat **voorbeeld.php** er als volgt uitziet:

```
<!DOCTYPE html>
<html lang="nl">
<head>
<title>Mijn php-script</title>
</head>
<body>
<h3>arrays</h3>
<?php
$weekdag = array("maandag", "dinsdag", "woensdag",
    "donderdag", "vrijdag", "zaterdag", "zondag");
echo("De eerste dag is: " . $weekdag[0]);
?>
</body>
</html >
```

4.4.3 Associatieve hash-arrays

In PHP kun je je eigen index associëren met een element. Een hash-array codeer je met de volgende syntaxis:

```
$prijs = array("melk"=>2.0, "koffie"=>3.50, "suiker"=>2);
```

Of

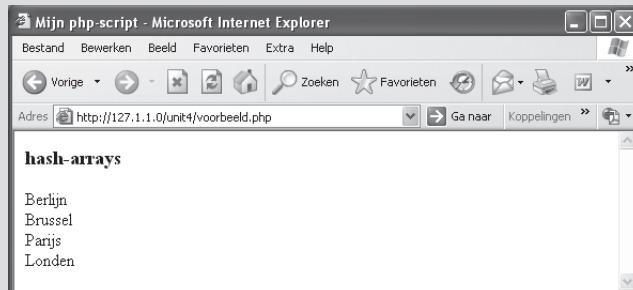
```
$prijs["melk"] = 2.0;
$prijs["koffie"] = 3.50;
$prijs["suiker"] = 2;
```

Bijvoorbeeld: de volgende echo-opdracht zal de prijs (**2.0**) die geassocieerd is met **melk** weergeven:

```
echo($prijs["melk"]);
```

- Opgave 3**
1. Verander bovenstaand **voorbeeld.php** in **les4.php**. Maak de volgende \$hoofdstad hash-array:
- DE Berlijn
BE Brussel
FR Parijs
GB Londen
2. Vervolgens geef je alle array-elementen weer.

Figuur 4.10
Het resultaat van opgave 3



PHP code camp - Lab 2

Hash-arrays

In deze lab-opdracht maak je drie hash-arrays met gegevens voor drie laptops.

Merk	Model	Operating system	Voorraad	Prijs
Toshiba Satellite	A100	Windows XP	80	999
Acer Aspire	5732Z	Linux	0	888
HP	200X	Vista	50	777

Maak eerst de hash-array \$toshiba en geef deze weer zoals hieronder.

Stap 1: Typ het volgende php-script over en sla het op als **lab02.php**.

```
<h3>php lab 02</h3>
<?php
$toshiba["merk"] = " Toshiba Satellite ";
$toshiba["model"] = " A100 ";
$toshiba["os"] = " Windows XP ";
$toshiba["voorraad"] = 80;
$toshiba["prijs"] = 999;
$totaal = $toshiba["prijs"];
echo("<table border='1> " .
"<caption>
<strong>SML laptops</strong>
</caption>
<thead>
<tr><th>Merk</th><th>Model</th><th>Operating system
</th><th>Voorraad</th><th>Prijs</th></tr>
</thead>
<tbody>
<tr>
```

```

<td>" . $toshiba["merk"] . "</td>" .
"<td>" . $toshiba["model"] . "</td>" .
"<td>" . $toshiba["os"] . "</td>" .
"<td>" . $toshiba["voorraad"] . "</td>" .
"<td>" . $toshiba["prijs"] . "</td>" .
"</tr>
<tfoot>
<tr><td colspan='4'>Totaal</td><td>" . $totaal . "
</td></tr></tfoot></table>");
?>

```

Stap 2: Maak de nieuwe hash-array \$acer voor de Acer Aspire laptop.

Stap 3: Maak de nieuwe hash-array \$hp voor de HP laptop.

Stap 4: Reken het totaalbedrag van de drie laptops uit.

Stap 5: Het resultaat van **lab02.php** moet er als volgt uitzien.

Figuur 4.11

PHP-lab 2

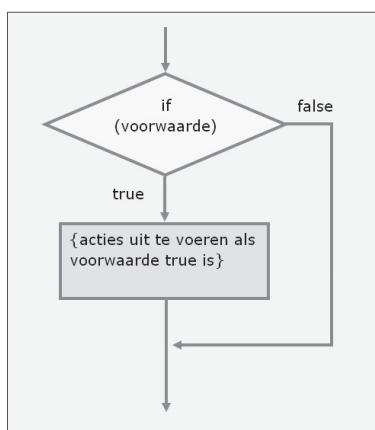
SML Laptops				
Merk	Model	Operating system	Voorraad	Prijs
Toshiba Satellite	A100	Windows XP	80	999
Acer Aspire	5732Z	Linux	0	888
HP	200X	Vista	50	777
Totaal				2664

4.5 De if-opdracht

Alle programmeertalen hebben beslissingsstructuren. Een beslissingsstructuur bepaalt de volgorde van uitvoering van de instructies in een programma. Je gebruikt de `if`-opdracht om te beslissen wat moet gebeuren als een voorwaarde `true` is.

Figuur 4.12

Het stroomdiagram van de if-opdracht



4.5.1 Syntaxis

De syntaxis van de `if`-opdracht in PHP is hetzelfde als in JavaScript:

```
if (voorwaarde)
{
    acties uit te voeren als voorwaarde true is;
}
```

4.5.2 De elseif-opdracht

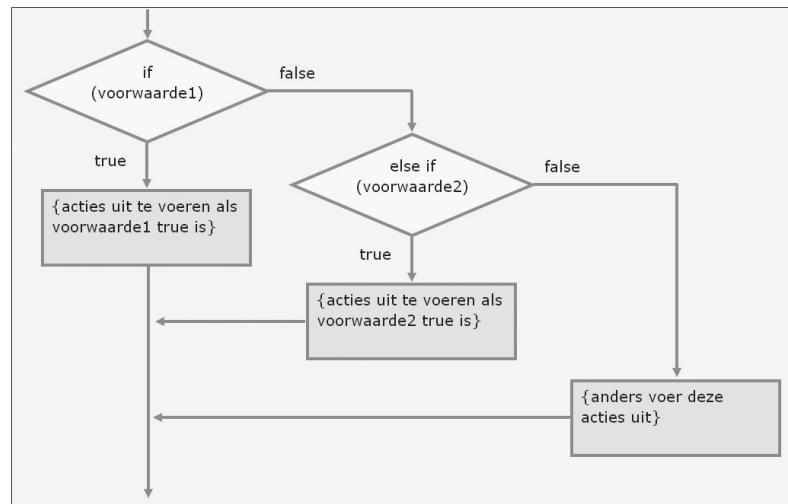
De `elseif`-opdracht geeft de mogelijkheid voor een alternatieve beslissing.

De syntaxis van de `if-elseif`-opdracht is als volgt:

```
if (voorwaarde1)
{acties uit te voeren als voorwaarde1 true is;}
else if (voorwaarde2)
{acties uit te voeren als voorwaarde2 true is;}
else
{anders deze acties uitvoeren;}
```

Figuur 4.13

Het stroomdiagram van de `if-elseif`-opdracht



De gebruikelijke vergelijkingsoperatoren zie je in de volgende tabel:

symbool	betekenis	voorbeeld
<code>==</code>	gelijk aan	<code>if(\$leeftijd == 16)</code>
<code>==</code>	identiek aan	<code>if(\$naam === " Carl")</code>
<code>!=</code> of <code><></code>	niet gelijk aan	<code>if(\$naam <> " Jan")</code>
<code>></code>	hoger dan	<code>if(\$leeftijd > 17)</code>
<code>>=</code>	hoger of gelijk aan	<code>if(\$a >= \$b)</code>
<code><=</code>	kleiner of gelijk aan	<code>if(\$a <= \$b)</code>
<code>&&</code>	AND	<code>if(\$a && \$b)</code>
<code> </code>	OR	<code>if(\$a = \$b \$a = \$c)</code>

Voorbeeld 4.5

Zorg ervoor dat **voorbeeld.php** er als volgt uitziet:

```
<!DOCTYPE html>
<html lang="nl">
<head>
<title>Mijn php-script</title>
</head>
<body>
<h3>de if opdracht</h3>
<?php
$gewerkteuren = 41;
$uurtarief = 15.00;
$bruto = $gewerkteuren * $uurtarief;
if($gewerkteuren > 40)
{
    $bonus = 90.00;
    echo("Uw salaris met bonus
        is:".€.".$bruto+$bonus));
}
else
{
    echo("Uw salaris is: " . "€" . $bruto);
}

?>
</body>
</html >
```

Figuur 4.14

Het resultaat van voorbeeld 4.5



4.5.3 De ternary-operator (?:)

De ternary-operator heeft de volgende syntaxis:

voorwaarde ? waarde als true : waarde als false

Bijvoorbeeld:

```
$module=($cijfer>=5 ? "voldoende" : "onvoldoende")
```

- Opgave 4** Verander bovenstaand **voorbeeld.php** in **les5.php**. Controleer met behulp van de ternary-operator of het salaris hoger dan 700 is. Als het salaris hoger dan 700 is, geef je de volgende melding weer: "U kunt maximaal 100 euro's sparen" Anders geef je de volgende melding weer: "U kunt maximaal 50 euro's sparen"

PHP code camp - Lab 3

De ternary-operator (? ... :)

De ternary-operator werkt zoals de opdracht `if-else`, bijvoorbeeld:

```
$leerplicht = false;
echo($leerplicht ? "leerplicht" : "niet leerplicht");
```

Stap 1: Open **lab02.php** en sla het als **lab03.php** op.

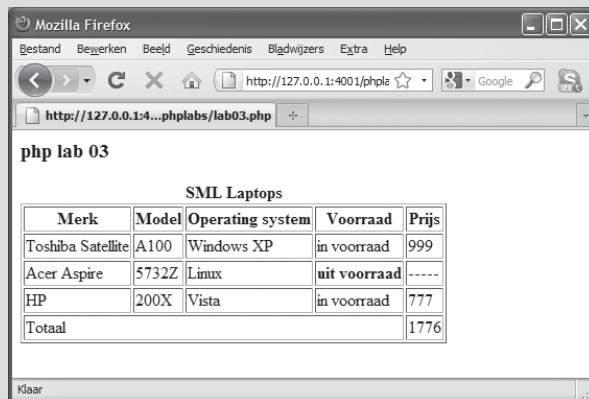
Stap 2: Zorg ervoor dat als de voorraad van de laptops nul is, dan de tekst "**uit voorraad**" wordt weergeven. Als de voorraad groter is dan nul, moet je de tekst "**in voorraad**" weergeven. Hiervoor gebruik je de ternary-operator.

Stap 3: Als de laptop uit voorraad is, dan geef je streepjes weer in plaats van de prijs en de prijs wordt niet mee geteld in het totaalbedrag.

Stap 4: Het resultaat van **lab03.php** moet er als volgt uitzien.

Figuur 4.15

PHP lab 3



4.6 Variabelen uit een formulier

Gebruikers kunnen een html-formulier gebruiken om gegevens in te voeren en te versturen. Een formulier wordt gedefinieerd door middel van de `<form>` `</form>`-tags.

4.6.1 De `<form>`-tag

Syntaxis:

```
<form action="verwerken.php" method="post">
```

- `action=`: dit attribuut geeft de URL van de pagina waar het formulier naartoe wordt gestuurd.
- `method=`: dit moet GET of post zijn.

Voorbeeld 4.6

Zorg ervoor dat **voorbeeld.php** er als volgt uitziet:

```
<!DOCTYPE html>
<html lang="nl">
<head>
<title>Mijn php-script</title>
</head>
```

```
<body>
<h3>Voorbeeld van een formulier</h3>
<form action="verwerken.php" method="post">
Uw naam : <input type="text" name="naam">
<p>
<input type="hidden" name="taal" value="false">
Kies een taal
<input type="radio" name="taal" value="N"> Nederlands
<input type="radio" name="taal" value="E"> Engels
<input type="radio" name="taal" value="S"> Spaans
</p>
<input type="submit" value="Versturen">
</form>
</body>
</html >
```

Figuur 4.16
Het resultaat van
voorbeeld 4.6



De drie keuzerondjes (radiobuttons) krijgen dezelfde naam (taal):

```
<input type="radio" name="taal" value="N">
<input type="radio" name="taal" value="E">
<input type="radio" name="taal" value="S">
```

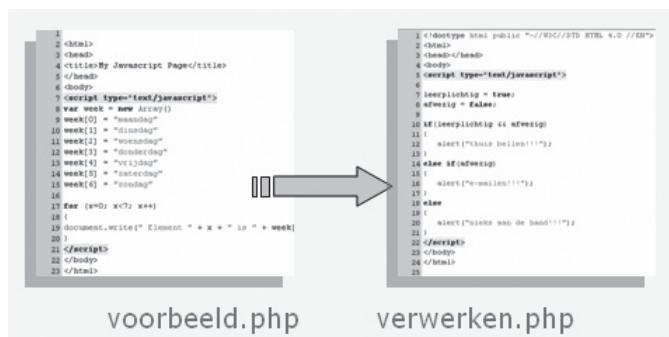
4.6.2 Gegevens verwerken

Kijk nogmaals naar de <form>-tag:

```
<form action="verwerken.php" method="post">
```

Als actie hebben we gekozen voor `action="verwerken.php"` en de methode om dat te doen is `method="POST"`. Dus als je op de versturen-knop klikt, zullen de twee variabelen `naam` en `taal` verstuurd worden naar het `verwerken.php`-script.

Figuur 4.17
Formuliergegevens worden naar het verwerken-script overgezet.



4.6.3 \$_POST-variabelen

In het **verwerken.php**-script heten deze variabelen:

```
$ _ POST["naam"] en $ _ POST["taal"]
```

In het verwerken-script kun je controleren of een variabele is ingevuld. Daarvoor gebruik je de `isset()`-functie, bijvoorbeeld:

```
isset($ _ POST["naam"])
```

Deze functie geeft de waarde `true` als **naam** in het formulier is ingevuld.

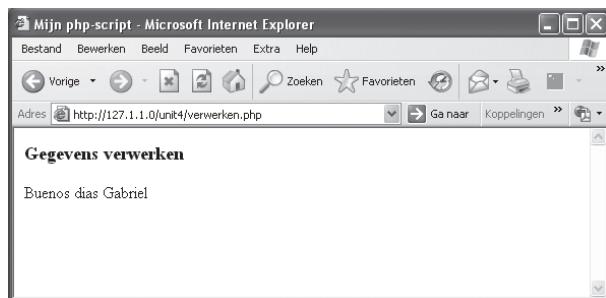
Voorbeeld 4.7

Zorg ervoor dat **verwerken.php** er als volgt uitziet:

```
<!DOCTYPE html>
<html lang="nl">
<head>
<title>Mijn php-script</title>
</head>
<body>
<h3>Gegevens verwerken</h3>
<?php
if(isset($ _ POST['naam']))
{
    if($ _ POST["taal"]=="N")
    {
        echo " Goedendag " . $ _ POST["naam"];
    }
    else if($ _ POST["taal"]=="E")
    {
        echo "Good morning " . $ _ POST["naam"];
    }
    else if($ _ POST["taal"]=="S")
    {
        echo "Buenos dias " . $ _ POST["naam"];
    }
}
else
{
    echo "naam of taal niet ingetypt";
}
?>
</body>
</html >
```

Als je de naam Gabriel intypt en op het keuzerondje voor Spaans klikt, zullen de gegevens worden verwerkt zoals in figuur 4.18.

Figuur 4.18
Het resultaat van voorbeeld 4.7



Het **voorbeeld.php**-script heeft de gegevens naar het **verwerken.php**-script verstuurd en het **verwerken.php**-script heeft de gegevens verwerkt.

Opgave

- 5 1. Verander bovenstaande **verwerken.php** in **les6.php**, zodat als er geen taal gekozen wordt, je de volgende melding toont:
U moet een taal kiezen xxxxxxxx
op de x's moet de ingetikte naam verschijnen.
2. Maak een nieuw script (**PO1inschrijven.php**) met daarin het formulier uit figuur 4.19.

Figuur 4.19

Het inschrijfformulier moet er als volgt uitzien

Maak een tweede script (**PO1verwerken.php**) dat de gegevens vanuit het formulier verwerkt zoals in figuur 4.20.

- Op de xxxx's moeten de gegevens uit het formulier verschijnen.
- Op de puntjes moet de naam van de gekozen opleiding verschijnen.
- Als gekozen werd voor ICT, toon je bovendien de volgende melding:
ICT-opleidingen zijn vol. Kies een andere opleiding.

Figuur 4.20

Het resultaat van het verwerken-
script



PHP code camp - Lab 4

Post-gegevens

In dit code-lab ‘posten’ we gegevens uit een formulier naar een php-script. Typ het volgende over en sla het als **lab04.php** op

```
<html xmlns="http://www.w3.org/1999/xhtml" lang="nl">
<head>
<title>XXL Computer winkel</title>
</head>
<body>
<h3>php lab 04</h3>
<table border=0 cellpadding=0 cellspacing=0 width=100%>
<form name="orderform" action="php echo $_SERVER['PHP_SELF']; ?" method="post">
<tr>
<td>

</td>
</tr>
<tr>
<td>
Toshiba Satellite A100-510    Basisprijs 999.99
</td>
</tr>
<tr>
<td><!--Shopping Cart Begin-->
<input type="hidden" name="toshibaproduct" value="001" />
<input type="hidden" name="toshibamerk" value="Toshiba" />
<input type="hidden" name="toshibamodel" value="Satellite A100-510" />
Aantal: <input type="text" size=2 maxlength=3 name="toshibaantal" value="0" />
<input type="hidden" name="toshibaprijs" value="999.99" />

```

```

<input type="image" src="bestel.jpg" border=0
       value="bestellen" />
<hr />
</td><!--Shopping Cart End -->
</tr>
</form>
</table>
</body>
</html>

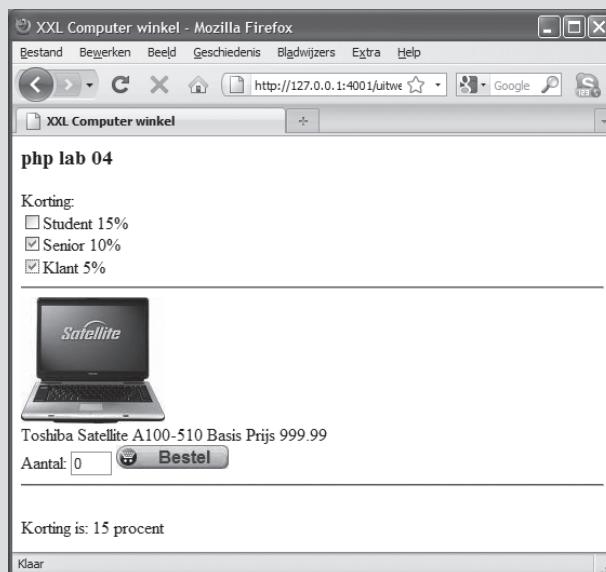
```

Stap 1: Codeer drie selectievakjes voor korting zoals hieronder.

Stap 2: Reken de totaalkorting uit.

Stap 3: Het resultaat moet er als volgt uitzien:

Figuur 4.21
PHP-lab 4



4.7 Switch

De PHP `switch`-opdracht kan efficiënter zijn dan de `if`-opdracht. We gebruiken de `switch`-opdracht wanneer we een blok uit veel blokken code moeten selecteren en uitvoeren.

4.7.1 Syntaxis

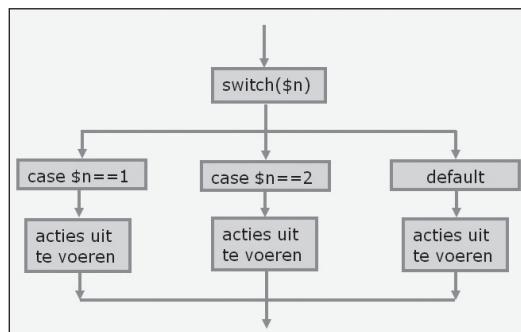
```

switch($n)
{
    case 1:
        acties uit te voeren als $n==1;
        break;
    case 2:
        acties uit te voeren als $n==2;
        break;
    default:
        acties uit te voeren als $n anders is dan 1 en 2;
}

```

Figuur 4.22

Het stroomdiagram van de switch-opdracht

**Voorbeeld 4.8**

Zorg ervoor dat **voorbeeld.php** er als volgt uitziet:

```

<!DOCTYPE html>
<html lang="nl">
<head>
<title>Mijn php-script</title>
</head>
<body>
<h3>Voorbeeld van de switch-opdracht</h3>

<form action=<?php echo $_SERVER['PHP_SELF']; ?>
method="post">
<p>Select een plaats:</p>
<select name="Plaats" value="true">
<option value=""></option>
<option value="a">Almere</option>
<option value="b">Amstelveen</option>
<option value="c">Amsterdam</option>
<option value="d">Bussum</option>
<option value="e">Hilversum</option>
<option value="f">Huizen</option>
<option value="g">Laren</option>
<option value="h">Weesp</option>
</select>
<p><input type="submit" name="versturen" value="Versturen"></p>
<p>-----</p>
</form>
<?php
if(isset($_POST['Plaats']))
{
    switch($_POST['Plaats'])
    {
        case "a" :
            echo "<p>U heeft Almere gekozen</p>";
            break;
        case "b" :
            echo "<p>U heeft Amstelveen gekozen</p>";
            break;
        case "c" :
    }
}
  
```

```

        echo "<p>U heeft Amsterdam gekzen</p>";
        break;
    case "d" :
        echo "<p>U heeft Bussum gekozen</p>";
        break;
    }
}
?>
</body>
</html >
```

- Kijk naar de inhoud van bovenstaand script. Binnen de `<form>`-tag zie je de volgende PHP-instructie:

`<?php echo $_SERVER['PHP_SELF']; ?>`

- De `$_SERVER['PHP_SELF']`-opdracht geeft aan dat de formuliergegevens in hetzelfde script verwerkt kunnen worden.

Opgave

- 6 Verander bovenstaand **voorbeeld.php** in **les7.php**, zodat als plaats niet gelijk aan a, b, c, of d is, geef je de volgende melding weer: **U moet een plaats kiezen.**

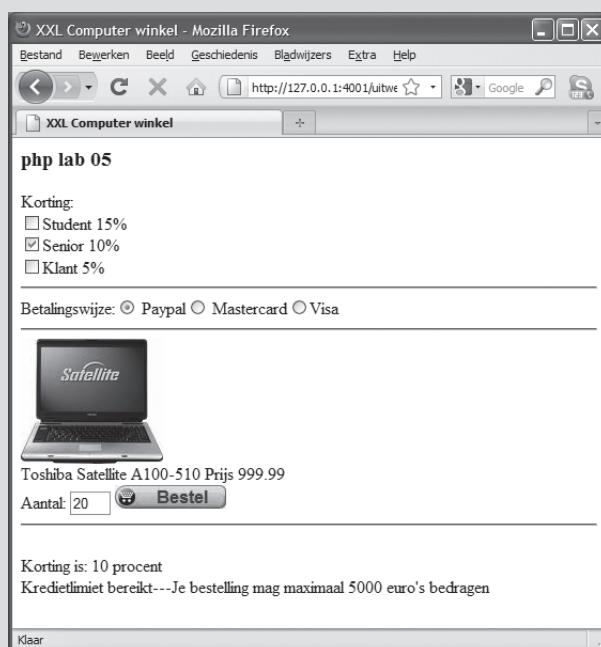
PHP code camp - Lab 5

De opdracht switch

In dit lab gaan we de opdracht `switch` gebruiken voor het kiezen van de betaalwijze.

Figuur 4.23

PHP-lab 5



Stap 1: Open **lab04.php** en sla het op als **lab05.php**.

Stap 2: Codeer drie keuzerondjes met betalingswijze, zoals in figuur 4.23.

Stap 3: Als het totaalbedrag van een bestelling groter dan is 5000 euro, toon je de volgende tekst:

Kredietlimiet bereikt---Je bestelling mag maximaal 5000 euro's bedragen

Stap 4: Controleer dat de betalingswijze is gekozen.

Stap 5: Maak een switch-opdracht voor betalingswijze. Er zijn drie 'cases':

Case 1: als de betalingswijze **PayPal** is, toon je de volgende tekst:

Uw betaling wordt behandeld via PayPal.

Case 2: als de betalingswijze **Mastercard** is, toon je de volgende tekst:

Uw betaling wordt behandeld via Mastercard.

Case 3: als de betalingswijze **Visa** is, toon je de volgende tekst:

Uw betaling wordt behandeld via Visa.

Stap 6: Het resultaat moet er als volgt uitzien:

4.8 Functies

4.8.1 Wat is een functie?

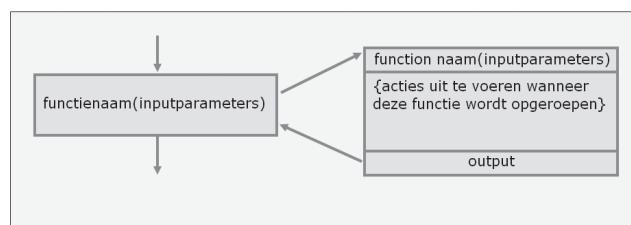
Een functie is code die een specifieke taak uitvoert. Functies codeer je apart. Dan zijn ze leesbaarder en je kunt ze hergebruiken wanneer je dezelfde taak moet uitvoeren.

In PHP declareer je een functie als volgt:

```
function functienaam ( $input-parameters )
{
    acties uit te voeren;
    wanneer deze functie wordt opgeroepen;
    met return($output) heb je de optie om output te
genereren;
}
```

Figuur 4.24

Stroomdiagram van een functie



Een functie roep je als volgt op:

```
functienaam ( $input-parameters );
```

Voorbeeld 4.9

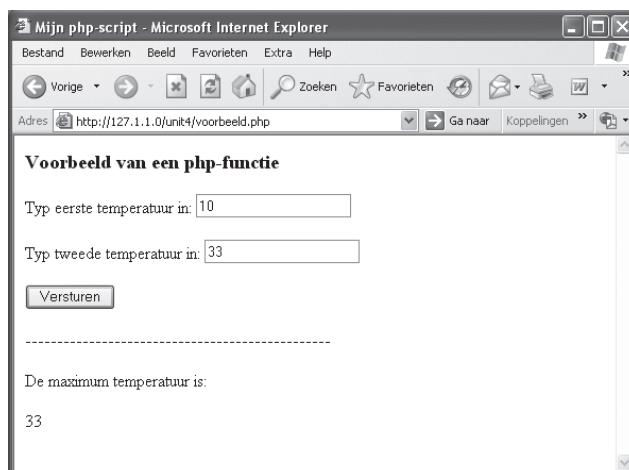
Zorg ervoor dat **voorbeeld.php** er als volgt uitziet:

```
<!DOCTYPE html>
<html lang="nl">
<head>
<title>Mijn php-script</title>
</head>
<body>
<h3>Voorbeeld van een php-functie</h3>
```

```
<?php
function maxGetal($getal1, $getal2)
{
    if($getal1 > $getal2)
    {
        return($getal1);
    }
    elseif($getal2 > $getal1)
    {
        return($getal2);
    }
    else
    {
        return("gelijk");
    }
}
?>
<form action=<?php echo $_ SERVER['PHP _ SELF']; ?>
method="post">
<p>Typ eerste temperatuur in: <input type="text"
name="temp1"></p>
<p>Typ tweede temperatuur in: <input type="text"
name="temp2"></p>
<p><input type="submit" name="versturen"
value="Versturen"></p>
<p>-----</p>
</form>
<p>De maximum temperatuur is:
<?php
if(isset($_ POST['temp1'])&& isset($_ POST['temp2']))
{
    echo maxGetal($_ POST['temp1'],$_ POST['temp2']);
}
?>
</p>
</body>
</html >
```

Figuur 4.25

Het resultaat van voorbeeld 4.9



Opgave

- 7 1. Verander bovenstaande **voorbeeld.php** in **les8.php**.
2. Codeer een nieuwe `minimaal()`-functie die het kleinste van twee getallen uitrekent.
3. In de `<body>`-sectie vervang je `maxGetal()`- met de `minimaal()`-functie.

PHP code camp - Lab 6**PHP-functies**

Een functie is als een zwarte doos. De acties tussen de accolades worden binnen de zwarte doos uitgevoerd.

Functie met input

Hier maken we een nieuwe functie met de naam **print** en de parameter **\$input**. Dit is de signatuur van de functie. In deze functie gooi je wat input in de zwarte doos. De zwarte doos kan dan iets met de input doen:

```
function print($input)
{
    echo($input);
}
```

**Figuur 4.26**

Een functie met input

Functie met input en output

Hier volgt een voorbeeld van een functie met input en output. Deze functie gebruikt de input om een resultaat te genereren en vervolgens om het resultaat te retourneren:

```
function verdubbelen($input)
{
    $output = $input*2;
    return($output);
}
```

**Figuur 4.27**

Een functie met input en output

Een functie aanroepen

Om een functie te kunnen gebruiken ('aanroepen') moet je de juiste signatuur van de functie gebruiken. De signatuur van een functie is de functienaam plus de functieparameters. Deze functie *verdubbelen* heeft als input een getal nodig, of een variabele met een getal.

**Figuur 4.28**

Een functie aanroepen.

Lab opgave

Stap 1: Typ het volgende php-script over en sla het op als **lab06.php**.

```
<h3>php lab 06 </h3>
<h4>Reis kosten berekenen </h4>
<?php
function reiskosten($vertrek, $bestemming)
{
    $reiskosten = array();
    $reiskosten[1] = array();
    $reiskosten[2] = array();
    $reiskosten[3] = array();
    $reiskosten[4] = array();
    $reiskosten[1][1] = 0;
    $reiskosten[1][2] = 30;
    $reiskosten[1][3] = 60;
    $reiskosten[1][4] = 90;

    echo($reiskosten[$vertrek][$bestemming] . " Euro's ");
}
?>
```

Stap 2: De reiskostenarray heeft twee indices: de eerste index voor vertrek en de tweede index voor bestemming. Index[1] is Amsterdam, index[2] is Utrecht, index[3] is Den Haag, index[4] is Rotterdam. Maak de array \$reiskosten af door de codes voor de volgende reiskosten in te voeren.

Tabel 4.1
De reiskosten-
array

Reis kosten	Amsterdam	Utrecht	Den Haag	Rotterdam
Amsterdam	0	30	60	90
Utrecht	30	0	40	20
Den Haag	60	40	0	10
Rotterdam	90	20	10	0

Stap 3: Maak de volgende interface voor de functie reiskosten:

Figuur 4.29
PHP-lab 6

php lab 06

Reiskosten

Vertrek: Amsterdam Bestemming: Rotterdam

Berekenen

De berekende reiskosten: 90 Euro's

Klaar

4.9 Externe functies

Je codeert PHP-functies binnen een PHP-script. Maar je kunt ook een externe functie coderen. Externe functies kun je vanuit een PHP-script oproepen met de `include()`- of `require()`-opdracht.

4.9.1 De `include()`- en `require()`-opdracht

Syntaxis:

```
include(bestandnaam.php)
```

of

```
require(bestandnaam.php)
```

Met `include()` en `require()` kun je bestanden met functies, tekst en html-tags opladen in je PHP-script. Het verschil tussen deze twee opdrachten is dat als een script niet gevonden wordt, geeft de `require()`-opdracht een foutmelding. De `include()`-opdracht geeft geen melding. Een externe `include` is ideaal wanneer je herbruikbare code wilt programmeren.

Voorbeeld 4.10

Zorg ervoor dat **mijnfuncties.php** er als volgt uitziet:

```
function maxGetal($getal1, $getal2)
{
    if($getal1 > $getal2)
    {
        return($getal1);
    }
    elseif($getal2 > $getal1)
    {
        return($getal2);
    }
    else
    {
        return("gelijk");
    }
}
```

Voorbeeld 4.11

Zorg ervoor dat **voorbeeld.php** er als volgt uitziet:

```
<!DOCTYPE html>
<html lang="nl">
<head>
<title>Mijn php-script</title>
</head>
<body>
<h3>Voorbeeld van include</h3>
<?php
```

```
include("mijnfuncties.php");
?>
<form action=<?php echo $_SERVER['PHP_SELF']; ?>
method="post">
<p>Typ eerste temperatuur in: <input type="text"
name="temp1"></p>
<p>Typ tweede temperatuur in: <input type="text"
name="temp2"></p>
<p><input type="submit" name="versturen"
value="Versturen"></p>
<p>-----</p>
</form>
<p>De maximum temperatuur is:
<?php
if(isset($_POST['temp1']) && isset($_POST['temp2']))
{
    echo maxGetal($_POST['temp1'],$_POST['temp2']);
}
?>
</p>
</body>
</html>
```

- Opgave 8**
- Codeer de nieuwe functie `max3()` in **mijnfuncties.php** die het hoogste van drie getallen uitrektent.
 - Verander bovenstaande **voorbeeld.php** in **les9.php** en voer de `max3()`-functie uit.

4.9.2 De `date()`-functie

In PHP kun je verschillende `date()`-functies gebruiken. Meer over de functie `date()` vind je in paragraaf 4.14.

Voorbeeld 4.12

Zorg ervoor dat **voorbeeld.php** er als volgt uitziet:

```
<!DOCTYPE html>
<html lang="nl">
<head>
<title>Mijn php script</title>
</head>
<body>
<?php
echo date("D")."<br />";
// geeft de weekdag weer (Mon, Tue..)
echo date("H:i:s") . "<br />";
// geeft het uur weer (13:42:08)
?>
</body>
</html>
```

Opgave 9 1. Verander bovenstaand **voorbeeld.php** in **PO2.php**.

2. Declareer de volgende \$weekdag-array:

```
$weekdag["Mon"] = "maandag";
$weekdag["Tue"] = "dinsdag";
$weekdag["Wed"] = "woensdag";
$weekdag["Thu"] = "donderdag";
$weekdag["Fri"] = "vrijdag";
$weekdag["Sat"] = "zaterdag";
$weekdag["Sun"] = "zondag";
```

3. Geef een welkomstmelding die gebaseerd is op de dag vandaag.

Figuur 4.30

Het resultaat van opgave 9 zou er zo kunnen uitzien



PHP code camp - Lab 7

Externe functies

In dit lab gaan we werken met een externe functie.

Stap 1: Codeer een externe functie die de beschikbaarheid van de laptop met de gekozen operating system retourneert.

	Besturingssysteem		
	XP	Vista	Linux
Toshiba	true	false	true
Acer	true	true	true
HP	true	false	False

Typ het volgende over en sla het als **bestellingfuncties.php** op.

```
<?php
function beschikbaarheid($merk, $os)
{
    // deze functie heeft twee input parameters
    // $merk verwijzt naar de merk-hash-array
    // $os verwijzt naar de os-hash-array
    // deze functie geeft de beschikbaarheid van de
    // laptop met os terug.
    $beschikbaar = array(
        "Toshiba" => array(
```

```

"xp" => true, "vista" => false, "linux" => true),
"Acer" => array(
"xp"=> true, "vista" =>true, "linux" => true),
"Hp" => array(
"xp"=> true,"vista" => false, "linux" => false)
);
return($beschikbaar[$merk][$os]);
}
?>

```

Stap 2: Open **lab05.php** en sla het als **lab07.php** op.

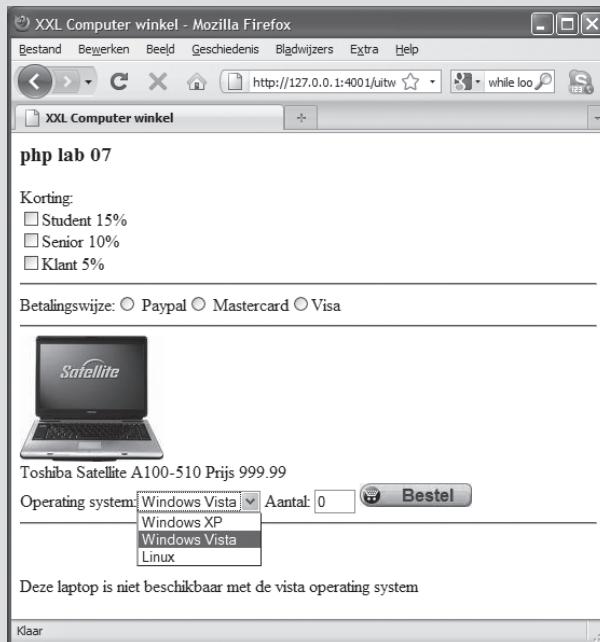
Stap 3: include("bestellingfuncties.php") in je script lab07.php

Stap 4: Roep de functie beschikbaarheid() aan. Geef de parameters voor het merk en het besturingssysteem aan.

Stap 5: Als de laptop met het gekozen besturingssysteem niet beschikbaar is, dan 'echo' je een melding zoals hieronder weergegeven.

Figuur 4.31

PHP-lab 7



4.10 De for-lus

Controlestructuren bepalen de volgorde van de uitvoering van de instructies in een programma. Bijvoorbeeld met de `for`-lus kun je een aantal keer hetzelfde blok instructies herhalen.

4.10.1 Syntaxis

De syntaxis van de PHP `for`-lus is hetzelfde als de JavaScript `for`-lus:

```

for($x=0;$x<6;$x++)
{
    acties uit te voeren zolang $x<6;
}

```

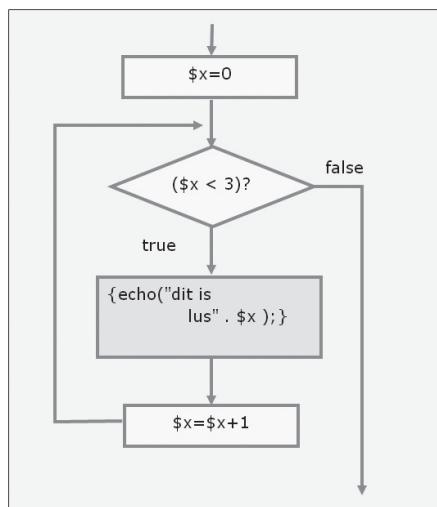
De volgende `for`-lus herhaalt de `echo()`-opdracht drie keer:

```
for ($x=0; $x<3; $x++)
{
echo("dit is lus" . $x);
}
```

- `$x=0` geeft aan dat de beginwaarde van de variabele `x` nul is.
- `$x<3` is de voorwaarde. Zolang deze voorwaarde `true` is, wordt de lus herhaald.
- Als de voorwaarde `false` is, wordt de lus beëindigd.
- `$x++`, de variabele `$x` wordt bij elke lus één opgehoogd.
- Acties uit te voeren: `echo("dit is lus " . $x);`

Figuur 4.32

Het stroomdiagram van de for-lus



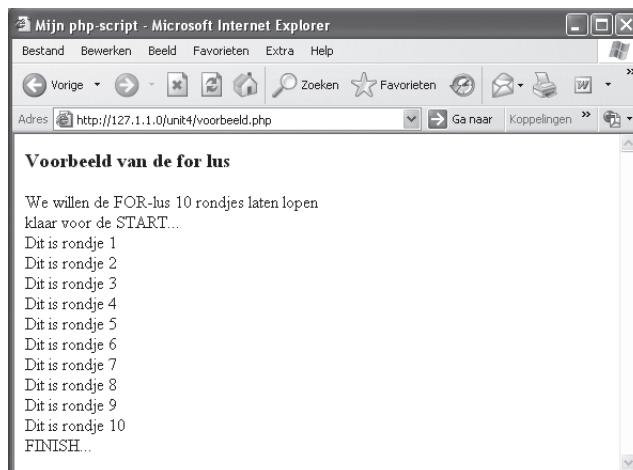
Voorbeeld 4.13

Zorg ervoor dat **voorbeeld.php** er als volgt uitziet:

```
<!DOCTYPE html>
<html lang="nl">
<head>
<title>Mijn php script</title>
</head>
<body>
<h3>Voorbeeld van de for lus</h3>
<?php
echo "We willen de FOR-lus 10 rondjes laten lopen
<br />";
echo "klaar voor de START...<br />";
for($steller=1; $steller<=10; $steller++)
{
    echo "Dit is rondje $steller <br />";
}
echo "FINISH...";
?>
</body>
</html>
```

Figuur 4.33

Het resultaat van voorbeeld 4.13



Opgave 10 1. Verander bovenstaand **voorbeeld.php** in **les12.php**.

2. Codeer een `for`-lus die de inhoud van de volgende `weekdag[]`-array weergeeft.

```

$weekdag[0] = "maandag";
$weekdag[1] = "dinsdag";
$weekdag[2] = "woensdag";
$weekdag[3] = "donderdag";
$weekdag[4] = "vrijdag";
$weekdag[5] = "zaterdag";
$weekdag[6] = "zondag";

```

PHP code camp - Lab 8

De for-lus

Stap 1: Open **lab07.php** en sla het op als **lab08.php**.

Stap 2: Codeer de functie bestellingoverzicht zodat de bestelling in een tabel wordt weergegeven zoals hieronder. Open **bestellingfuncties.php** en voeg de volgende functie eraan toe.

```

<?php
function bestellingoverzicht()
{
$info[1] = "product";
    $info[2] = "merk";
    $info[3] = "model";
    $info[4] = "os";
    $info[5] = "aantal";
    $info[6] = "prijs";

    echo "<br />
<table width='80%' border='1' >
<caption>
<strong>Bestellingoverzicht</strong>
</caption>
<thead>

```

```

<tr><th>Product</th><th>Merk</th><th>Model
</th><th>OS</th><th>Aantal</th><th>Basisprijs</th>
</tr>
</thead>
<tbody>";

$reij = "<tr>";
for($x=1; $x<=sizeof($info); $x++)
{
    $reij = $reij . "<td>" .
    $_POST["toshiba.$info[$x]"] . "</td>";
}
$reij = $reij . "</tr>";
echo $reij;
echo "</tbody></table>";
}
?>

```

Stap 3: Maak deze functie af zodat er een bestellingoverzicht in een tabelformaat wordt weergegeven. Het resultaat moet er als volgt uitzien:

Figuur 4.34
PHP-lab 8

Korting:
 Student 15%
 Senior 10%
 Klant 5%

Betalingswijze: Paypal Mastercard Visa

Toshiba Satellite A100-510 Basisprijs 999.99
 Operating system: Aantal:

Bestellingoverzicht					
Product	Merk	Model	OS	Aantal	Basisprijs
001	Toshiba	Satellite A100-510	xp	3	999.99

Korting is: 10 procent
 Uw bestelling wordt behandeld via Visa

Klaar

4.11 De while-lus

In de `while`-lus testen we voor een voorwaarde. De voorwaarde kan waar of onwaar zijn. Bijvoorbeeld: `is ($totaal == 100) ?`

Als de voorwaarde **waar** is, wordt de code tussen de accolades uitgevoerd. Als de voorwaarde **niet waar** is, wordt de code tussen de accolades **niet meer** uitgevoerd.

4.11.1 Syntaxis

```
while ( voorwaarde )
{
    acties uit te voeren;

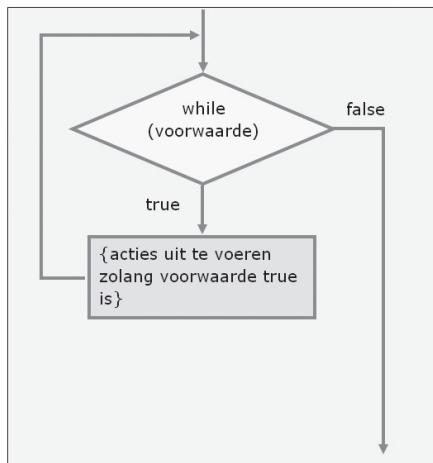
}
```

Voorbeeld 4.14

Zorg ervoor dat **voorbeeld.php** er als volgt uitziet:

```
<!DOCTYPE html>
<html lang="nl">
<head>
<title>Mijn php script</title>
</head>
<body>
<h3>Voorbeeld van de while-lus</h3>
```

Figuur 4.35
Stroomdiagram
van de while-lus



```
<?php
echo "Zolang ons saldo lager is dan 2000
willen we dat de while-lus ons maandelijks saldo
uitrekent<br />";
$saldo = 750;
$rente = 0.1;
$maand = 1;
echo "Begin saldo is:" . $saldo . "<br />";
echo "START... <br />";
while($saldo < 2000)
{
    $saldo = $saldo + ($saldo * $rente);
    echo "Maand " . $maand . " is je saldo: " . $saldo . "<br
/>";
    $maand++;
}
echo "FINISH";
?>
</body>
</html>
```

Figuur 4.36

Het resultaat van voorbeeld 4.14

```
Zolang ons saldo lager is dan 2000 willen we dat de while-lus ons maandelijks saldo
uitrektent
Begin saldo is:750
START...
Maand 1 is je saldo: 825
Maand 2 is je saldo: 907.5
Maand 3 is je saldo: 998.25
Maand 4 is je saldo: 1098.075
Maand 5 is je saldo: 1207.8825
Maand 6 is je saldo: 1328.67075
Maand 7 is je saldo: 1461.537825
Maand 8 is je saldo: 1607.6916075
Maand 9 is je saldo: 1768.46076825
Maand 10 is je saldo: 1945.306845075
Maand 11 is je saldo: 2139.8375295825
FINISH
```

4.11.2 De do-while-lus

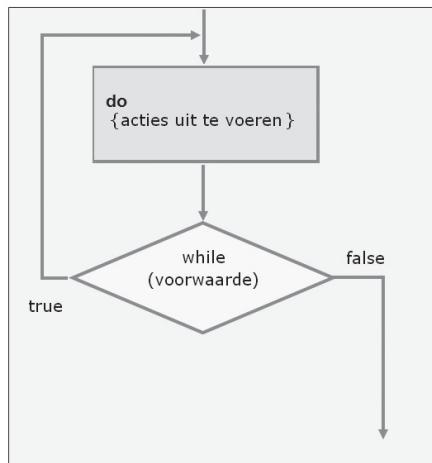
De do-while-lus is weer anders dan de while-lus. Het verschil is dat bij de do-while-lus de lus eerst wordt uitgevoerd, waarbij aan het einde van de eerste lus de voorwaarde wordt aangegeven voor eventuele herhaling. Dat betekent dat de do-while-lus altijd minimaal een keer wordt uitgevoerd.

Syntaxis

```
do
{
    acties uit te voeren;
}
while( voorwaarde );
```

Figuur 4.37

Stroomdiagram van de do-while-lus



Voorbeeld 4.15

Zorg ervoor dat **voorbeeld.php** er als volgt uitziet:

```
<!DOCTYPE html>
<html lang="nl">
<head>
<title>Mijn php-script</title>
</head>
<body>
<h3>Voorbeeld van de do-while-lus</h3>
<?php
echo "Reken het saldo uit. Daarna, zolang ons saldo lager is
dan 2000
    willen we de do-while-lus herhalen<br />";
$saldo = 2100;
$rente = 0.1;
$maand = 1;
echo "Begin saldo is:" . $saldo;
echo "<br>START...";
do
{
    $saldo = $saldo + ($saldo * $rente);
    echo "<br>Maand " . $maand . " je saldo is: " . $saldo;
    $maand++;
}
while($saldo < 2000);
echo "<br>FINISH";
?>
</body>
</html>
```

Figuur 4.38

Resultaat van voorbeeld 4.15



4.11.3 Een lus onderbreken

Er zijn drie opties voor het onderbreken van de normale uitvoering van een lus:

- Als je midden in een lus wilt stoppen met het uitvoeren van de lus, gebruik je de opdracht **break**. Met deze opdracht onderbreek je de lus maar je php-script gaat door met de volgende regel na de lus.

- Als je midden in een lus wilt beginnen met de volgende iteratie (herhaaling) van de lus, gebruik je de opdracht **continue**. Met deze opdracht wordt de huidige iteratie onderbroken, maar de lus gaat door naar het begin van de volgende iteratie.
- Als je wilt stoppen met het uitvoeren van het hele php-script, gebruik je de opdracht **exit**. Met deze opdracht wordt het uitvoeren van je php-script onderbroken.

PHP code camp - Lab 9

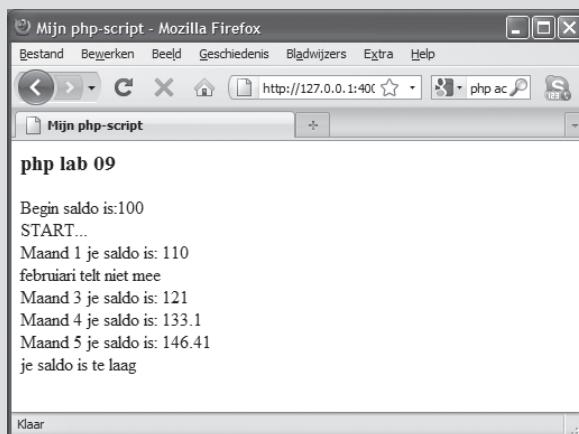
De while-lus

Open **voorbeeld52.2.php** en sla het als op **lab09.php**. In deze lab-opdracht onderbreek je de do-while-lus als volgt:

- Als \$maand gelijk is aan 2, geef je de melding ‘februari telt niet mee’; daarna onderbreek je de huidige iteratie en begin je met de volgende iteratie.
- Als het huidige \$saldo boven de 2000 ligt, geef je de melding ‘Maximale saldo 2000 is bereikt’; daarna onderbreek je de hele lusopdracht.
- Als \$maand gelijk is aan 6 en \$saldo is onder de 1000, geef je de melding ‘Je saldo is te laag’; daarna onderbreek je het script.

Als je beginsaldo 100 is, moet het resultaat er als volgt uitzien:

Figuur 4.39
lab 9



4.12 De foreach-lus

Soms wil je de elementen uit een array herhaaldelijk verwerken. De **foreach-lus** maakt het eenvoudig om met arrays te werken. De **foreach-lus** haalt elke keer een element vanuit je array.

4.12.1 Syntaxis

```

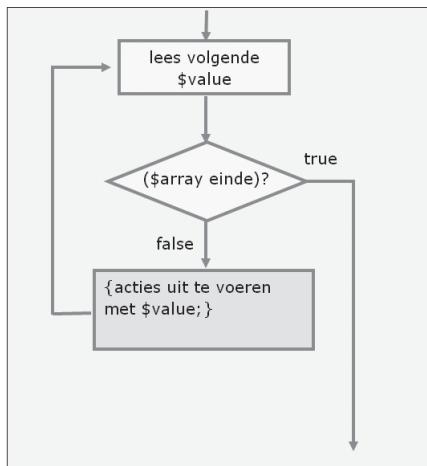
foreach ($arraynaam as $value)
{
    acties uit te voeren;
}

```

Neem als voorbeeld de volgende array. Je kunt met de `foreach`-lus elke keer een element vanuit de array lezen en specifieke acties uitvoeren. Als er geen elementen meer te lezen zijn, wordt de `foreach`-lus beëindigd.

\$arraynaam		
[0]	[1]	[2]
\$value1	\$value2	\$value3

Figuur 4.40
Stroomdiagram
van de foreach-lus



Voorbeeld 4.16

Zorg ervoor dat **voorbeeld.php** er als volgt uitziet:

```

<!DOCTYPE html>
<html lang="nl">
<head>
<title>Mijn php script</title>
</head>
<body>
<h3>Voorbeeld van de foreach-lus</h3>
<?php
$weekdag[0] = "Zondag";
$weekdag[1] = "Maandag";
$weekdag[2] = "Dinsdag";
$weekdag[3] = "Woensdag";
$weekdag[4] = "Donderdag";
$weekdag[5] = "Vrijdag";
$weekdag[6] = "Zaterdag";
foreach($weekdag as $value )
{
    echo "$value <br />";
}

?>
</body>
</html>
  
```

Figuur 4.41

Het resultaat van voorbeeld 4.16



4.12.2 Foreach-lus met hash-array

Een hash-array kun je ook met de `foreach`-lus verwerken.

Syntaxis:

```
foreach ($arraynaam as $key => $value)
{
    acties uit te voeren;

}
```

Voorbeeld 4.17

Zorg ervoor dat **voorbeeld.php** er als volgt uitziet:

```
<!DOCTYPE html>
<html lang="nl">
<head>
<title>Mijn php script</title>
</head>
<body>
<h3>Voorbeeld van de foreach-lus met $key</h3>
<?php
$hoofdstad["DE"] = "Berlijn";
$hoofdstad["BE"] = "Brussel";
$hoofdstad["ES"] = "Madrid";
$hoofdstad["DK"] = "Kopenhagen";
$hoofdstad["FR"] = "Parijs";
$hoofdstad["GB"] = "London";
foreach($hoofdstad as $key => $value )
{
    echo "$key hoofdstad is $value <br />";
}

?>
</body>
</html>
```

Figuur 4.42

Het resultaat van voorbeeld 4.17



- Opgave 11** 1. Verander bovenstaande **voorbeeld.php** in **les14.php**.
2. Codeer onderstaande hash-array:

oranje	orange
rood	red
paars	violet
groen	green
blauw	blue
geel	yellow

3. Codeer een `foreach`-lus die de volgende tekst weergeeft: `echo "
dezelfde tekst in $key"`

Figuur 4.43

Het resultaat van opgave 11 moet er zo uitzien



PHP code camp - Lab 10

De foreach-lus

In dit lab ga je een toepassing coderen van de `foreach`-lus.

*Stap 1: Open **lab08.php** en sla het op als **lab10.php**.*

*Stap 2: Codeer een functie bestellingoverzicht zodat een bestelling met meerdere laptops in een tabel wordt weergegeven zoals hieronder. Open **bestellingfuncties.php** en voeg de volgende functie eraan toe.*

```
<?php
function bestellingoverzicht()
{
    $laptop["toshiba"] = "toshiba";
    $laptop["acer"] = "acer";

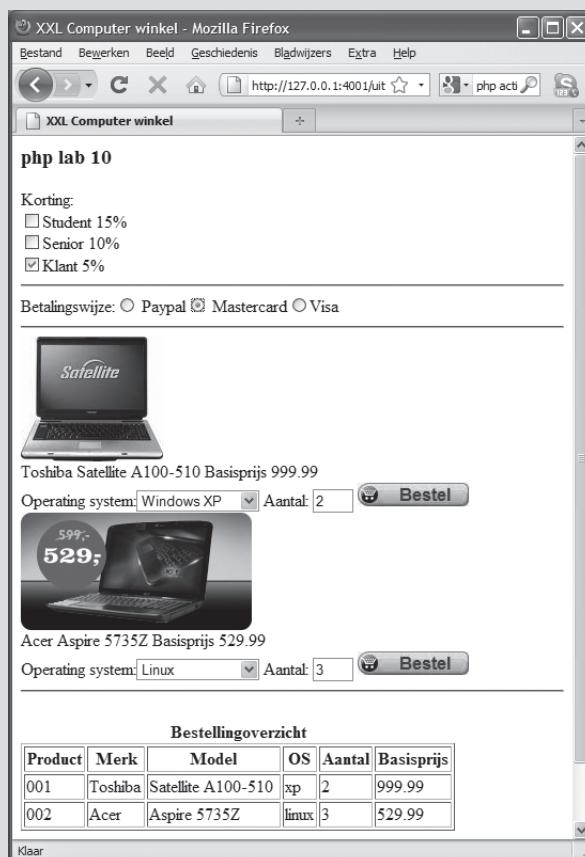
    $info[1] = "product";
    $info[2] = "merk";
    $info[3] = "model";
    $info[4] = "os";
    $info[5] = "aantal";
    $info[6] = "prijs";

    echo "<br />
<table width='80%' border='1' >
<caption>
<strong>Bestellingoverzicht</strong>
</caption>
<thead>
<tr><th>Product</th><th>Merk</th><th>Model
</th><th>OS</th><th>Aantal</th><th>Basisprijs</th>
</tr>
</thead>
<tbody>";

    foreach($laptop as $key => $value)
    {
        $aantal = $value."aantal";
        if($_POST[$aantal] == 0) continue;
        $reij = "<tr>";
        for($x=1; $x<=sizeof($info); $x++)
        {
            $element = $value . $info[$x];
            $reij = $reij . "<td>" . $_POST[$element] . "</
td>";
        }
        $reij = $reij . "</tr>";
        echo $reij;
    }
    echo "</tbody></table>";
}
?>
```

Stap 3: Open **lab10.php** en voeg er een nieuwe laptop aan toe, zodat er een overzicht als een tabel wordt weergegeven. Het resultaat moet er als volgt uitzien:

Figuur 4.44
PHP-lab 10



4.13 Constanten en globale variabelen

Iedere programmeertaal werkt met het begrip ‘scope’. De scope of reikwijdte van een variabele geeft aan waar in een script die variabele ‘zichtbaar’ is. Php heeft ook regels voor de scope van variabelen. Bijvoorbeeld, variabelen kunnen globale of lokale variabelen zijn.

Superglobale variabelen

Er zijn ingebouwde superglobale variabelen, die overal in je script zichtbaar zijn. Een paar voorbeelden:

- `$GLOBALS` is een array met alle globale variabelen. We kunnen een `$GLOBALS`-variabele vanuit een functie benaderen met:
`$GLOBALS['variabelenaam']`
- `$_GET` is een array met formuliervariabelen die je met `method="get"` zichtbaar maakt in je script.
- `$_POST` is een array met formuliervariabelen die je met `method= "post"` zichtbaar maakt in je script.
- `$_COOKIE` is een array met cookie-variabelen.
- `$_FILES` is een array met variabelen over file uploads.
- `$_SESSION` is een array met session-variabelen. Een session onstaat gedurende een bezoek aan je website.

Typ het volgende over en sla het op als **voorbeeld.php**.

```
<?
$GLOBALS['url'] = "www.mijndomeinnaam.nl";
//superglobale variabele

function show()
{
    echo "<br />URL:" . $GLOBALS['url'];
show();
?>
```

Een superglobale variabele is overal in het script, binnen en buiten je functies zichtbaar.

Globale variabelen

Een globale variabele is zichtbaar binnen het script maar niet binnen een functie. Open **voorbeeld.php** en voeg de volgende codes eraan toe:

```
<?
$GLOBALS['url'] = "www.mijndomeinnaam.nl";
//superglobale variabele
global $email;
$email = "webmaster@mijndomeinnaam.nl";
//globale variabele

function show()
{
    echo "<br />URL:" . $GLOBALS['url'];
}
show();
?>
```

De globale variabele `$email` is niet zichtbaar binnen de functie `show()`. Als je een globale variabele zichtbaar wilt maken binnen een functie, moet je de variabele binnen de functie weer als global declareren.

Open **voorbeeld.php** en voeg de volgende codes eraan toe:

```
<?
$GLOBALS['url'] = "www.mijndomeinnaam.nl";
//superglobale variabele
global $email;
$email = "webmaster@mijndomeinnaam.nl";
//globale variabele

function show()
{
    global $email;
    //maak de globale variabele hier zichtbaar
    echo "<br />URL:" . $GLOBALS['url'];
    echo "<br />email:" . $email;
}
show();
?>
```

Constanten

Een constante is het tegenovergestelde van een variabele: de waarde van een constante blijft altijd hetzelfde. Denk bijvoorbeeld aan gegevens die zelden veranderen, zoals het btw-percentage of een bedrijfsadres.

Open **voorbeeld.php** en voeg de volgende codes eraan toe:

```
<?
$GLOBALS['url'] = "www.mijndomeinnaam.nl";
// superglobale variabele
global $email;
$email = "webmaster@mijndomeinnaam.nl";
//globale variabele

define('ADRES', "Kruislaan 111"); //constante

function show()
{
    global $email;
    //maak de globale variabele hier zichtbaar
    echo "<br />URL:" . $GLOBALS['url'];
    echo "<br />email:" . $email;
}
show();
?>
```

Constanten zijn ook globaal zichtbaar binnen en buiten functies. De naam van een constante typen we in hoofdletters en zonder \$-teken. Zo kunnen we duidelijk zien dat het om een constant gaat.

Statische variabelen

Statische ('static') variabelen die gemaakt worden binnen een functie zijn niet zichtbaar buiten de functie. De waarde van een statische variabele binnen een functie wordt bijgehouden van de ene naar de volgende aanroep.

Open **voorbeeld.php** en voeg de volgende codes eraan toe:

```
<?
$GLOBALS['url'] = "www.mijndomeinnaam.nl";
// superglobale variabele
global $email;
$email = "webmaster@mijndomeinnaam.nl";
// globale variabele

define('ADRES', "Kruislaan 111"); //constante
define('BTW', 0.19); //constante

echo $email;

function show()
{
    global $email;
    //maak de globale variabele hier zichtbaar
    echo "<br />URL:" . $GLOBALS['url'];
```

```

        echo "<br />email:" . $email;
    }
    function showbezoekers()
    {
        static $aantalbezoekers;
        //static variabele binnen functie
        $aantalbezoekers++;
        echo "<br />Aantal Bezoekers:" . $aantalbezoekers;
    }

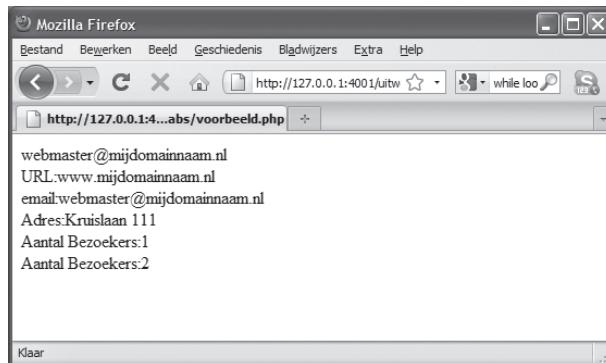
    show();
    showbezoekers();
    showbezoekers();

?>

```

Figuur 4.45

Het resultaat van voorbeeld.php



Hier is de waarde van de static variabele `$aantalbezoekers` opgeteld elke keer dat de functie `showbezoekers()` wordt aangeroepen.

Lokale variabelen die gemaakt zijn binnen een functie verdwijnen aan het einde van die functie.

PHP code camp - Lab 11

Constante en globale variabelen

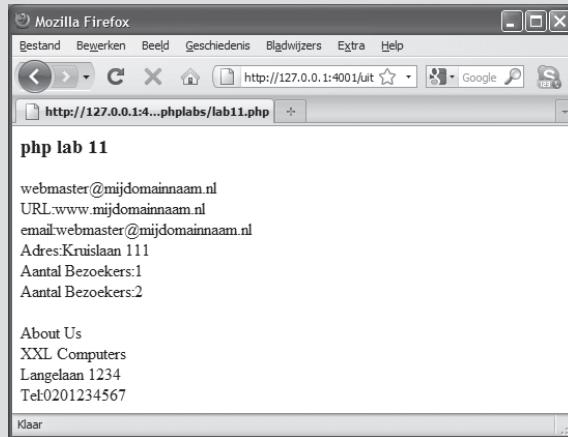
In dit lab gaan we een functie programmeren die een aantal bedrijfsgegevens toont. Open **voorbeeld.php** en sla het op als **lab11.php**. Voor de bedrijfsgegevens maak je gebruik van de volgende constanten en variabelen:

- de constante bedrijfsnaam = “XXL Computers”
- de constante bedrijfsadres = “Langelaan 1234”
- de globale variabele telefoonnummer = “0201234567”

Maak een functie `aboutus()` die deze bedrijfsgegevens toont. Het resultaat moet er als volgt uitzien.

Figuur 4.46

PHP-lab 11



4.14 De functie date()

De functie `date()` heeft twee input parameters. De eerste is een string voor het formatteren van de datum. De tweede parameter is een optionele Unix ‘timestamp’. Als deze tweede parameter niet wordt gespecificeerd, geeft de functie `date()` de huidige datum terug.

De formatting string

De functie `date()` met een formatting string ziet er bijvoorbeeld zo uit:

```
echo date('jS F Y');
```

Het resultaat is dat huidige datum als volgt geformatteerd wordt:

1 januari 2000

De `j` formateert de dag van de maand (1 t/m 31)

De `F` formateert de maand in letters (januari t/m december)

De `Y` formateert het jaar in vier cijfers (bijvoorbeeld 2010)

Je kunt uit verschillende codes kiezen om de datum naar wens te formatteren:

Code	Omschrijving
c	Datum in de vorm JJJJ-MM-DD T HH:MM:SS
d	Dag van de maand in twee cijfers (01 t/m 31)
j	Dag van de maand in cijfers (1 t/m 31)
D	Dag van de week in drie letters (Mon t/m Sun)
G	Uur in de vorm (0 t/m 23)
H	Uur in de vorm (00 t/m 23)
m	De maand in twee cijfers (01 t/m 12)
M	De maand in drie letters (jan t/m dec)

Code	Omschrijving
F	De maand in letters (januari t/m december)
w	Dag van de week in de vorm (0 t/m 6) voor zondag/zaterdag
y	Jaartal in twee cijfers (bv. 11)
Y	Jaartal in vier cijfers (bv. 2011)

Typ het volgende over en sla het op als **voorbeeld.php**.

```
<?php
$datum = date("d F y ");
echo("<br />Geformatterd:" . $datum);
?>
```

time()

De Unix timestamp is een 32-bit integer met het aantal seonden die verlopen zijn sinds middernacht 1 januari 1970. De functie `time()` maakt een conversie van de datum naar een Unix timestamp. Open **voorbeeld.php** en voeg de volgende codes eraan toe:

```
<?php
$timestamp = time();
echo("<br />Timestamp:" . $timestamp);
?>
```

mktime()

De functie `mktime()` maakt een conversie van de datum naar een Unix timestamp als volgt: `mktime(uur, minuut, seconde, maand, dag, jaar)`. Open **voorbeeld.php** en voeg de volgende codes eraan toe:

```
<?php
$dag = 23;
$datum = mktime(12,0,0,0,$dag + 30);
echo("<br />over 30 dagen:" . $datum );
?>
```

De `getdate()`-function

Deze functie krijgt een input timestamp en geeft een array terug met de volgende elementen:

Key	Value
Seconds	Seconden
Minutes	Minuten
Hours	Uur
Mday	Dag van de maand (getal)
Wday	Dag van de week (getal)
Mon	Maand (getal)

Key	Value
Year	Jaar
Weekday	Dag van de week (tekst)
Month	Maand (tekst)

Open **voorbeeld.php** en voeg de volgende codes eraan toe:

```
<?php
$datum = mktime();
$nu = getdate($datum);
echo("<br />Tijd:" . $nu["hours"] . ":" . $nu["minutes"] .
":" . $nu["seconds"]);
?>
```

4.15 Cookies?

Cookies zijn kleine bestandjes, die opgeslagen worden op de harde schijf van de bezoeker. Hierin kan informatie opgeslagen worden of de bezoeker al eerder op de website is geweest. Wanneer de browser een verbinding maakt met een URL, zoekt de browser naar bestaande cookies op de hardeschijf van de bezoeker. Als de cookies bestaan, zijn deze cookies beschikbaar in de array `$_COOKIES[]`.

4.15.1 Cookies plaatsen

Je kunt een cookie plaatsen op de computer van je bezoeker met de volgende code:

```
setcookie("naam", "inhoud");
```

Typ het volgende over en sla het op als **voorbeeld.php**.

```
<?php
setcookie("naam", "inhoud");
?>
```

Bij *naam* vul je de naam van het cookie in, *inhoud* vervang je door de inhoud van het cookie. Om een cookie te maken met een geldigheidsduur van een uur gebruik je de volgende code:

```
setcookie("gebruiker", "sanskrit", time() + 3600);
```

Typ het volgende over en sla het op als **voorbeeld.php**.

```
<?php
setcookie("gebruiker", "sanskrit", time() + 3600);
?>
```

Hierin staat 3600 voor het aantal seconden, in dit geval dus een uur (60×60 s = 3600 s). Je kunt ook een datum opgeven waarop het cookie verloopt; dat doe je bv met de volgende code:

```
setcookie("gebruiker", "sanskrit", mksime(0,0,0,1,1,2010));
```

Typ het volgende over en sla het op als **voorbeeld.php**.

```
<?php
setcookie("gebruiker", "sanskrit", mkttime(0,0,0,1,1,2010));
?>
```

4.15.2 Cookies lezen

Om de inhoud van een eerder geplaatste cookie weer oproepen gebruik je de volgende code:

```
echo $_COOKIE["gebruiker"];
```

Typ het volgende over en sla het op als **voorbeeld.php**.

```
<?php
setcookie("gebruiker", "sanskrit", mkttime(0,0,0,1,1,2010));
echo $_COOKIE["gebruiker"];
?>
```

4.15.3 Cookies verwijderen

De inhoud van een cookie kan gewijzigd worden door simpelweg een cookie met dezelfde naam, maar andere inhoud eroverheen te schrijven. Als je een cookie hebt geplaatst met een geldigheidsduur, dan kun je de cookie ook verwijderen door de geldigheidsduur negatief te maken. Typ het volgende over en sla het op als **voorbeeld.php**.

```
<?php
setcookie("gebruiker", "sanskrit", mkttime(0,0,0,1,1,2011));
echo $_COOKIE["gebruiker"];
setcookie("gebruiker", "sanskrit", mkttime(0,0,0,1,1,2007));
?>
```

PHP code camp - Lab 12

Cookies

In deze lab-opdracht maak een nieuw cookie met als naam je eigen naam. De inhoud van dit cookie is een teller die het aantal keer bijhoudt dat je op bezoek ben geweest op deze webpagina. Typ het volgende over en sla het op als **lab12.php**.

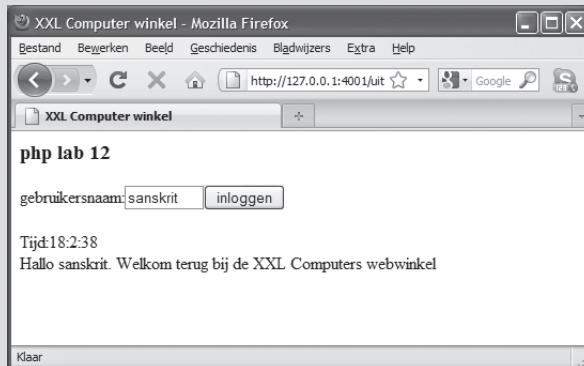
```
<!DOCTYPE html>
<html lang="nl">
<head>
<title>XXL Computer winkel</title>
</head>
<body>
<h3>php lab 12</h3>
<table border=0 cellpadding=0 cellspacing=0 >
<form name="orderform" action=<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
<tr>
<td>gebruikersnaam:</td>
<td><input type="text" name="user" /></td>
```

```
<td><input type="button" name="send" value="inloggen" /></td>
</tr>
</form>
</table>
<?php
include("cookiefuncties.php");
welkom();
?>
</body>
</html>
```

Maak een nieuw script en sla het op als **cookiefuncties.php**. Maak een nieuwe functie `welkom()` zodat er wordt gezocht of het gebruikerscookie bestaat. Als de gebruikers cookie gelijk is aan de inlog gebruikersnaam dan geef je de melding: ‘Hallo “gebruikersnaam”, welkom terug bij de XXL Computers website.’

Anders maak je een gebruikerscookie met de inlognaam van de huidige gebruiker en geef je de melding: ‘Welkom “gebruikersnaam”, dit is je eerste bezoek.’

Figuur 4.47
PHP-lab 12



4.16 Reguliere expressies

We kunnen string-functies en reguliere expressies gebruiken voor het scannen van woorden of subteksten in een string. Deze functies en reguliere expressies zijn handig voor het verifiëren van input gegevens van een klant. Net als in JavaScripts gebruiken we in PHP ook reguliere expressies voor het matchen van patronen in strings.

Patronen

De punt(.) wordt in PHP in reguliere expressies gebruikt voor het specificeren van een ‘wild-card teken’. Bijvoorbeeld, de expressie:

.uis

komt overeen (‘matcht’) onder andere met de volgende strings: muis, huis, luis, 2uis, 9uis, *uis. Kortom, alle woorden die beginnen met een willekeurige teken en eindigen met uis zijn een match.

Met reguliere expressies kun je ook een specifiek teken of een groep tekens aangeven. Bijvoorbeeld, een specifiek teken vanaf **a** tot en met **z** geef je tussen vierkante haakjes aan:

[a-z]uis

Deze expressie matcht met alle woorden die beginnen met een teken van a t/m z en eindigen met uis: uis, buis, cuis, huis t/m zuis. Er zijn nog meer mogelijkheden:

[a-zA-Z]	Beschrijft een kleine letter of een hoofdletter.
[^a-zA-Z]	Beschrijft een teken dat geen kleine letter a t/m z mag zijn.
*	Het symbool ^ binnen vierkante haakjes betekent ‘geen.’
*	Geeft aan dat het patroon mag nul of meer malen herhaald mag worden.
+	Geeft aan dat het patroon mag een of meer malen herhaald mag worden.

Bijvoorbeeld:

[a-zA-Z] *	Beschrijft een of geen kleine letter of hoofdletter.
[a-zA-Z] +	Beschrijft een of meer kleine letters of hoofdletters; dit komt dus neer op ‘minimaal een letter.’

Subexpressies

Een expressie kun je in delen declareren. Bijvoorbeeld een URL bestaat uit vier delen: protocol, www, domeinnaam, en com. Een subexpressie geef je aan tussen rondehakjes. Bijvoorbeeld `(www)` betekent dat www is maar een deel van een expressie.

Subexpressie teller

Het aantal keer dat een subexpressie voorkomt geef je tussen accolades {} aan. Bijvoorbeeld `(w){1,3}` betekent dat de letter w mag een of twee of drie keer voorkomen.

Begin van expressie(^)

Met het symbool ^ geef je het begin van een expressie aan. Bijvoorbeeld `^http` matcht http aan het *begin* van een string.

Einde van expressie(\$)

Met het symbool \$ geef je het einde van een expressie aan. Bijvoorbeeld `com$` matcht com aan het *einde* van de string.

Meerdere keuzes(|)

Met het symbool | geef je meerdere keuzes aan. Bijvoorbeeld `nl|com|net` matcht nl of com of net.

Optioneel(?)

Om een deel van je expressie optioneel te declareren gebruik je het symbool ?. Bijvoorbeeld `^(http://)?(www)$` betekent dat het eerste deel optioneel is.

Speciale tekens

Als je een van de speciale tekens `\$.|()^*+{}?` wilt matchen, dan gebruik je eerst het backslash teken (\). Bijvoorbeeld de expressie voor de string `^$` geef je als volgt aan: `\^ \$`

Expressie voor e-mailadres

Met behulp van deze mogelijkheden kun je ingewikkelde expressies opbouwen. De expressie voor een e-mailadres ziet er bijvoorbeeld als volgt uit:

1. Het eerste deel bestaat uit een combinatie van alfanumeriek tekens: `^[a-zA-Z0-9_\-.]+`
2. Dan komt het @ symbool: `^[a-zA-Z0-9_\-.]+\@[a-zA-Z0-9_\-.]+`
3. Daarna volgt een combinatie van alfanumerieke tekens en streepjes: `^[a-zA-Z0-9_\-.]+\@[a-zA-Z0-9_\-.]+\.`
4. Dan een punt: `^[a-zA-Z0-9_\-.]+\@[a-zA-Z0-9_\-.]+\.`
5. En het eindigt met een combinatie van alfanumerieke tekens, streepjes en punten: `^[a-zA-Z0-9_\-.]+\@[a-zA-Z0-9_\-.]+\.\[a-zA-Z0-9_\-.]+\$`

De functie eregi()

De functie `eregi()` matcht een reguliere expressie met een string. Zo kunnen we controleren of een bepaalde string geformuleerd is volgens de reguliere expressie. De syntaxis van de functie `eregi()` is als volgt:

```
eregi("reguliereexpressie" . $string)
```

Als de string geformuleerd is volgens de reguliere expressie dan is het resultaat van de functie true, anders is het false.

PHP code camp - Lab 13

Reguliere expressies

Open **lab10.php** en sla het als **lab13.php** op. Voeg de volgende formulier-codes eraan toe:

```
<form name="klantgegevens" action="<?php  
echo $_SERVER['PHP_SELF']; ?>" method="post">  
<table border=1 >  
<tr>  
    <td colspan="2">  
        <b>Factuur klantgegevens</b>  
    </td>  
</tr>  
<tr>
```

```
<td width="100">Naam: </td>
<td>
    <input type="text" size="55" name="naam" />
</td>
</tr>
<tr>
    <td>Adres:</td>
    <td>
        <input type="text" size="55" name="adres">
    </td>
</tr>
<tr>
    <td>Woonplaats:</td>
    <td>
        <input type="text" size="34" name="woonplaats">
        Postcode:<input type="text" size="6" name="postcode">
    </td>
</tr>
<tr>
    <td>e-mail:</td>
    <td>
        <input type="text" size="55" name="email">
    </td>
</tr>
<tr>
    <td>Feedback:</td>
    <td>
        <textarea cols="40", rows="3" name="commentaar">
        </textarea>
    </td>
</tr>
</table>
<input type="image" src="checkout.jpg" value="send"/>
</form>
```

Figuur 4.48
Factuur
klantgegevens

Product	Merk	Model	OS	Aantal	Basisprijs
001	Toshiba	Satellite A100-510	xp	3	999.99
002	Acer	Aspire 5735Z	linux	2	529.99

Typ het volgende over en sla het als **formfuncties.php** op.

```
<?
$_email = $_POST['email'];
$_naam = $_POST['naam'];
$_adres = $_POST['adres'];
$_plaats = $_POST['plaats'];
$_postcode = $_POST['postcode'];
$_commentaar = $_POST['commentaar'];

function emailcheck()
{
    global $_email;
    if(eregi('^[a-zA-Z0-9 _\-.]+@[a-zA-Z0-9 _\-.]+\.[a-zA-Z]{2,3}$' , $_email))
    {
        echo("<br />email adres is correct");
        return(true);
    }
    else
    {
        echo("<br />email adres niet geldig");
    }
}
```

```

        return(false);
    }

}

?>

```

Open **formfuncties.php** en codeer de functie `postcodecheck()`. Deze functie controleert met een reguliere expressie of de postcode uit het klantgegevensformulier correct is. (vier cijfers plus twee letters). Het resultaat moet er bijvoorbeeld als volgt uitzien:

Figuur 4.49

E-mail en postcode
gecontroleerd

Product	Merk	Model	OS	Aantal	Basisprijs
001	Toshiba	Satellite A100-510	xp	2	999.99
002	Acer	Aspire 5735Z	xp	3	529.99

4.17 String functies

trim()

De functie `trim()` verwijdert spaties vanuit het begin en einde van een input string. Bijvoorbeeld `trim(" Umut ")` geeft als resultaat "Umut".

Je kunt ook de functies `ltrim()` en `rtrim()` gebruiken voor het verwijderen van spaties links of rechts van een string. Open **formfuncties.php** en wijzig de volgende regels zoals hieronder:

```

$naam = trim($_POST['naam']);
$adres = trim($_POST['adres']);

```

```
$plaats = trim($_POST['plaats']);
$postcode = trim($_POST['postcode']);
$email = trim($_POST['email']);
$commentaar = trim($_POST['commentaar']);
```

nl2br()

De functie `nl2br()` gebruiken we voor het formatteren van tekst naar hypertekst of html-tekst. Dit gaat op de volgende manier: alle nieuwe-regeltekens ('newlines') worden vervangen door de html-code `
`. Zo kunnen we de lange string vanuit het element `$_POST['commentaar']` formatteren voor het weergeven in een browser.

Open **formfuncties.php** en voeg de volgende regel in zoals hieronder:

```
$commentaar = nl2br($commentaar);
```

strtoupper()

De functie `strtoupper()` verandert een input string in hoofdletters. Bijvoorbeeld `strtoupper("amsTERdam")` geeft als resultaat `"AMSTERDAM"`.

Open **formfuncties.php** en voeg de volgende functie eraan toe:

```
function bezorgkosten()
{
    global $plaats;
    $plaats = strtoupper($plaats);
    if($plaats == "AMSTERDAM")
    {
        $bezorgkosten = 10.00;
    }
    elseif($plaats == "UTRECHT")
    {
        $bezorgkosten = 20.00;
    }
    else
    {
        $bezorgkosten = 30.00;
    }
}
```

strtolower()

De functie `strtolower()` verandert een input string in kleine letters. Bijvoorbeeld `strtolower("LETTERS")` geeft als resultaat `"letters"`.

Open **formfuncties.php** en voeg de volgende regel eraan toe:

```
$email = strtolower($email);
```

ucfirst()

De functie `ucfirst()` wijzigt de eerste letter vanuit een string in hoofdletter. Bijvoorbeeld

```
ucfirst("umut sandoval") geeft als resultaat "Umut Sandoval".
```

Open **formfuncties.php** en voeg de volgende regel eraan toe:

```
$naam = ucfirst($naam);
```

explode()

De functie `explode()` splitst een input string in substrings, gebaseerd op het scheidingsteken. Het resultaat is een array met de gesplitste substrings. Bijvoorbeeld, het splitsen van de string `$email` bij het teken `@` codeer je als volgt:

```
$email = "user@domain.com";
$earray = explode('@', $email);
// resultaat $earray[0] is "user"
// resultaat $earray[1] is "domain.com"
```

Open **formfuncties.php** en voeg de volgende codes eraan toe:

```
$emailarray = explode('@', $email);
$user = $emailarray[0];
$domain = $emailarray[1];
echo("<br />user:" . $user);
echo("<br />domain:" . $domain);
```

strlen()

De functie `strlen()` geeft als resultaat de lengte van de input string. Met deze functie kunnen we input gegevens controleren. Bijvoorbeeld:

```
$postcode = "1000XXXX";
$postcodelengte = strlen($postcode);
// resultaat is 8
```

Open **formfuncties.php** en voeg de volgende codes eraan toe:

```
if(strlen($postcode) != 6) echo("<br /> Postcode incorrect ingevuld.");
```

substr()

De functie `substr()` geeft als resultaat een deel van de input string. Deze functie krijgt drie input parameters: `substr(input string, begin positie in input string, lengte van de substring)`, bijvoorbeeld:

```
$postcode = "1000XX";
$postcodeprefix = substr($postcode, 0,4);
// resultaat is "1000"
```

Open **formfuncties.php** en voeg de volgende codes eraan toe:

```
$postcodeprefix = substr($postcode, 0,4);
$postcodesufix = substr($postcode,4,2);
echo("<br />postcodeprefix:" . $postcodeprefix);
echo("<br />postcodesufix:" . $postcodesufix);
```

strpos()

De functie `strpos()` geeft als resultaat de positie van een substring in de input string. Als de substring niet gevonden is, dan is het resultaat false.

Deze functie krijgt drie input parameters: `strpos(input_string, substring, [offset])`. Offset is optioneel en geeft de beginpositie in de inputstring, bijvoorbeeld:

```
$email = "user@domain.nl";
$n1 = strpos($email, ".nl");
// resultaat is "12"
```

Open **formfuncties.php** en voeg de volgende codes eraan toe:

```
$n1 = strpos($email, ".nl");
$be = strpos($email, ".be");
$fr = strpos($email, ".fr");
if($n1 > 0) echo("<br /> nationaliteit is nederland");
if($be > 0) echo("<br /> nationaliteit is belgisch");
if($fr > 0) echo("<br /> nationaliteit is frans");
```

str_replace()

De functie `str_replace()` gebruik je voor het zoeken en vervangen van substrings in een input string. Zo kunnen we een meer persoonlijk php-document genereren met het vervangen van bijvoorbeeld `<><>` door de persoonsnaam en `<><>` door het adres. De te vervangen woorden en de nieuwe woorden kun je ook in arrays doorgeven. Bijvoorbeeld:

```
$scheldwoorden = array("debiel", "laaf", "gestoord");
str_replace($scheldwoorden, "*#@#*!%!", $commentaar);
```

Het resultaat is dat alle scheldwoorden in `$commentaar` worden vervangen door `"*#@#*!%"`

Open **formfuncties.php** en voeg de volgende codes eraan toe:

```
$scheldwoorden = array("debiel", "laaf", "gestoord");
$commentaar = str_replace($scheldwoorden, "*#@#*!%!", $commentaar);
echo("<br /> gefilterd commentaar is:" . $commentaar);
```

PHP code camp - Lab 14

Stringfuncties

Open **formfuncties.php** en codeer de functie `commentaarevaluieren()`. Deze functie zoekt in het commentaar naar positieve en negatieve woorden. Als er meer positieve dan negatieve woorden voorkomen dan is de evaluatie positief. Het resultaat moet er als volgt uitzien:

Figuur 4.50
PHP-lab 14

4.18 Gegevensbestanden

Er zijn twee manieren om gegevens op te slaan: een bestand of een database zoals MySQL. Als je veel gegevens hebt, gebruik je een database. In dit geval gaan we de gegevens uit onderstaand aanmeldingsformulier in een bestand opslaan.

4.18.1 Gegevens opvragen

Opgave 12 Maak een nieuw PHP-script **aanmelden.php** met daarin het formulier uit figuur 4.51.

4.18.2 Gegevens opslaan

Gegevens in een bestand opslaan doen we in drie stappen:

- Stap 1 Bestand openen.
- Stap 2 Bestand schrijven.
- Stap 3 Bestand afsluiten.

Figuur 4.51

Het resultaat van opgave 12 moet er zo uitzien

Aanmeldingsformulier

Achternaam:	<input type="text"/>
Voornaam:	<input type="text"/>
Adres:	<input type="text"/>
Postcode:	<input type="text"/>
Plaats:	<input type="text"/>
email adres:	<input type="text"/>
Gebruikersnaam:	<input type="text"/>
Wachtwoord:	<input type="text"/>

4.18.3 Bestand openen met fopen()

PHP opent een bestand met de `fopen()`-functie. Als het bestand nog niet bestaat, wordt het bestand eerst aangemaakt en daarna geopend, bijvoorbeeld:

```
$bestand=fopen("gebruikers.txt","w");
```

De eerste parameter is de bestandsnaam. De tweede parameter is de open-mode. Een bestand open je om te schrijven, om te lezen of allebei. In de tabel staat een samenvatting van de open mode:

Mode	naam	betekenis
r	read	Open om te lezen.
r+	read/write	Open om te lezen en schrijven.
w	write	Open om te schrijven: deze mode begint altijd met een leeg bestand.
a	append	Open om toe te voegen: als bestand nog niet bestaat, maak een nieuw bestand.
b	binary	Bestandsformaat is binaire.
t	text	Bestandsformaat is tekst (alleen in Windows-systemen).

Als de `fopen()`-functie een bestand niet kan openen, geeft deze functie de waarde `false`. Dit kun je als volgt melden:

```
$bestand=fopen("gebruikers.txt","ab");
if(!$bestand)
{
    echo("kon geen bestand openen!");
}
```

4.18.4 Bestand schrijven met fwrite()

PHP schrijft een bestand met de `fwrite()`-functie bijvoorbeeld:

```
fwrite($bestand, $outputtekst, $lengte);
```

- `$outputtekst` is de tekst die je wilt schrijven.

- `$lengte` is het maximaal aantal bytes van de outputtekst. Bijvoorbeeld:

```
fwrite($bestand,$outputtekst,strlen($outputtekst));
```

De `strlen()`-functie geeft de lengte van een string-tekst. De outputtekst represeneert een rij (record) in je bestand, bijvoorbeeld:

```
$outputtekst =
$_POST["achternaam"] . $_POST["voornaam"] . "\t" .
$_POST["adres"] . "\t" .
$_POST["postcode"] . "\t" .
$_POST["plaats"] . "\t" .
$_POST["email"] . "\t" .
$_POST["gebruikersnaam"] . "\t" .
$_POST["wachtwoord"] . "\n";
```

- De `\t` is de code voor een tab en de `\n` voor een nieuwe regel.

4.18.5 Bestand afsluiten met fclose()

Nadat je een bestand hebt aangemaakt of gelezen, moet je het bestand altijd afsluiten. Dat doe je met de `fclose()`-functie, bijvoorbeeld:

```
fclose($bestand);
```

Deze functie geeft de waarde `true` bij het afsluiten van het bestand of `false` als het bestand niet afgesloten kan worden. Bijvoorbeeld:

```
if(fclose($bestand))
{
    echo("Account is aangemaakt");
} else {
    echo("Kon bestand niet afsluiten");
}
```

- Opgave 13** Maak een nieuw PHP-script **accountaanmaken.php** dat de gegevens uit het aanmeldingsformulier in een bestand opslaat en daarna de volgende melding weergeeft: **Account is aangemaakt**

Figuur 4.52

Het resultaat van opgave 13 moet er zo uitzien



Het resultaat in het **gebruikers.txt**-bestand zou er als volgt kunnen uitzien:

Hansen Ola	Timoteivn 10	1010BB	Amsterdam	ab@ab.nl	hano	wls
Svenson Tove	Borgvn 23	1099AQ	Utrecht	st@xl.nl	tovo	sax
Pettersen Kari	Storgt 20	1010DA	Amsterdam	pa@tx.nl	peto	axa

4.18.6 Bestand lezen

Om een bestand te kunnen lezen moet je eerst het bestand openen, bijvoorbeeld:

```
$bestand=fopen("gebruikers.txt","r");
if(!$bestand)
{
    echo("kon geen bestand openen!");
}
```

4.18.7 Bestand lezen met fgets()

Met **fgets()** lees je een hele rij gegevens. We willen de rijen (records) uit het **gebruikers.txt**-bestand lezen. Elke keer dat we een record lezen, gebruiken we de **feof()**-functie om te kijken of het einde van het bestand (end of file) bereikt is. Bijvoorbeeld:

```
while(!feof($bestand))
{
    $account = fgets($bestand, 999);
    echo $account . "<br>";
}
fclose($bestand);
```

De **fgets()** functie leest elke keer een rij

- tot de nieuwe rij (**\n**) code of
- tot een end of file (**EOF**) marker of
- tot het maximumaantal gespecificeerde bytes, in dit geval (999 – 1).

Opgave 14 Maak een nieuw PHP-script **accountweergeven.php** dat de gegevens uit het **gebruikers.txt**-bestand leest en weergeeft.

Figuur 4.53

Het resultaat van opgave 14 moet er als volgt uitzien



4.18.8 Bestand lezen met fgetcsv()

Om de individuele gegevens uit een rij (record) te kunnen lezen kopiëren we eerst de hele rij naar een array. Bijvoorbeeld:

```
$account = fgetcsv($bestand, 100, "\t");
```

De tweede parameter is de lengte en moet langer zijn dan de langste rij in het bestand.

De "\t" is het tabscheidingsteken tussen de array-elementen.

Bijvoorbeeld:

\$account						
[0]	[1]	[2]	[3]	[4]	[5]	[6]
Hansen Ola	Elderlaan10	1010BB	Amsterdam	ab@ab.nl	hano	wls

- Opgave 15**
- Maak een nieuw PHP-script **inloggen.php** dat de gebruikersnaam en het wachtwoord uit een inlogformulier leest.
 - Daarna controleer je of deze gebruikersnaam en dit wachtwoord in **gebruikers.txt** bestaan.

Figuur 4.54

Het resultaat van opgave 15 moet er als volgt uitzien

PHP code camp - Lab 15

In deze praktijkopdracht maak je een weblog waarin gebruikers hun eigen commentaren kunnen bloggen. Het thema bepaal je zelf. Om deze weblog te kunnen maken moet je sessions programmeren.

Wat zijn sessions?

Gedurende het browsen door verschillende webpagina's kun je gegevens genereren. Een session is een set van gegevens tussen de klant en de server. Deze gegevens kun je in andere webpagina's gebruiken zonder dat je de POST hoeft te gebruiken. Applicaties met inloggen-scripts, zoals onlinebanking en webwinkels, maken gebruik van sessions om gebruikersinteractie te kunnen managen.

We beginnen met het **inloggen.php** uit opgave 54D. Nadat de gebruiker ingelogd is, start je een session voor deze gebruiker zoals in dit **setup.php**-script:

```
// Verwerk de POST variabelen.  
$gebruiker = $_POST['naam'];  
  
// Start een session voor deze gebruiker.  
session_start();  
  
// Maak variabelen voor deze session.  
$_SESSION['klant'] = $gebruiker;  
$_SESSION['counter'] = 0;  
  
// Vraag de welkompagina op.  
header('Location: welkom.php');
```

Het setup-script is verantwoordelijk voor het maken van een nieuwe session. Een nieuwe session maak je met `session_start()`. In een session kunnen we session-variabelen opzetten (dat zijn globale variabelen die beschikbaar zijn gedurende de session). Nadat de session is gestart, wordt het **welkom.php**-script opgevraagd.

De welkompagina en alle andere webpagina's beginnen altijd met `session_start()`. Zo krijg je toegang tot de globale `$_SESSION`-variabelen. Hieronder zie je een codefragment van het **welkom.php**-script:

```
// Vind de session.  
session_start();  
  
// Welkom de klant.  
print "Hallo " . $_SESSION['klant'] . " welkom bij mijn  
applicatie";  
// .....  
  
// Session variabelen bijhouden.  
$_SESSION['counter']++;  
// .....  
  
// ga naar uitloggen script  
echo "<a href= 'uitloggen.php'> loguit </a>
```

Wanneer de gebruiker op loguit klikt, komt hij bij het **uitloggen.php**-script. Het logout-script moet ook, zoals alle andere scripts, met `session_start()` beginnen. Hier wordt de session verwijderd met `session_destroy()`. Hieronder zie je een codefragment van het **uitloggen.php**-script:

```
// Vind de session  
session_start();  
  
// Eind session melden.  
echo "Tot ziens " . $_SESSION['klant'] . " tot de volgende  
keer";  
// .....  
  
// Verwijder de session.  
session_destroy();
```

In deze praktijkopdracht maak je de volgende scripts:

- inloggen.php
- setup.php
- welkom.php
- blogpost.php
- uitloggen.php
- accounts.txt en blogs.txt

Figuur 4.55
Een inlogscherm



Maak nu Portfolio-opdracht 3 – Mailserver factureren

5

Databases

5.1 Inleiding databases

Databases zijn een essentieel onderdeel van de informatiemaatschappij. Een bedrijf of overheidsinstantie kan niet functioneren zonder databases. Om een database te bouwen begin je met een gegevensanalyse.

5.1.1 Wat is gegevensanalyse?

Gegevensanalyse stelt een organisatie in staat om interne en externe gegevens te analyseren en structureren. Deze analyses worden gebruikt voor de dagelijkse besturing van de organisatie en voor belangrijke besluitvorming binnen de organisatie.

5.1.2 Uitgangspunt

Als uitgangspunt nemen we het informatiesysteem voor een fictief bedrijf MP3shop. Zoals iedere organisatie draait MP3shop op informatiesystemen.

5.1.3 Top-downmodel

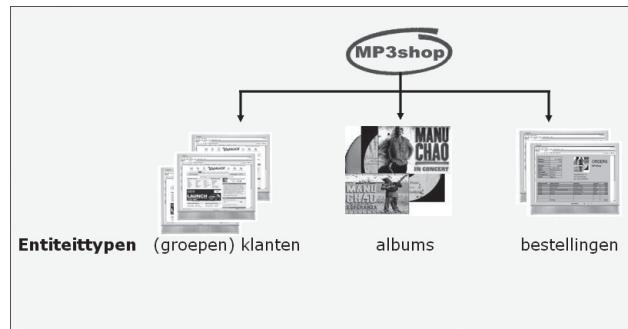
Voor de analyse van informatie van de MP3shop passen we een top-downmodel toe. Een top-downmodel gaat van globale naar gedetailleerde informatie. De elementen van een top-downmodel zijn:

- Entiteittypen
- Entiteiten
- Attributen
- Sleutels

5.1.4 Wat zijn entiteittypen?

Entiteittypen zijn groepen waarover dezelfde soort informatie wordt bijgehouden. Zie bijvoorbeeld figuur 5.1.

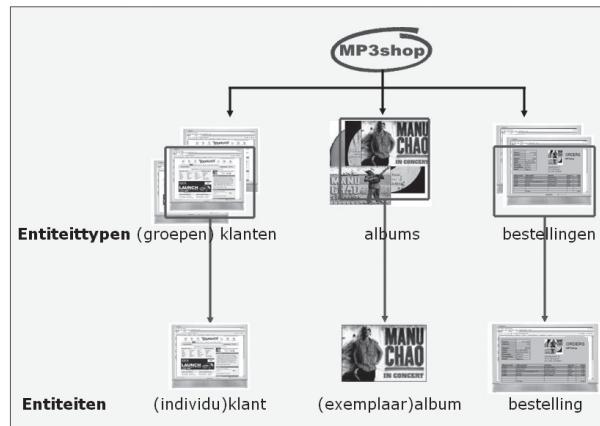
Figuur 5.1
Entiteittypen



5.1.5 Wat zijn entiteiten?

Een entiteit is een bepaald individu/exemplaar van een entiteitstype. Bijvoorbeeld een klant, een album of een bestelling.

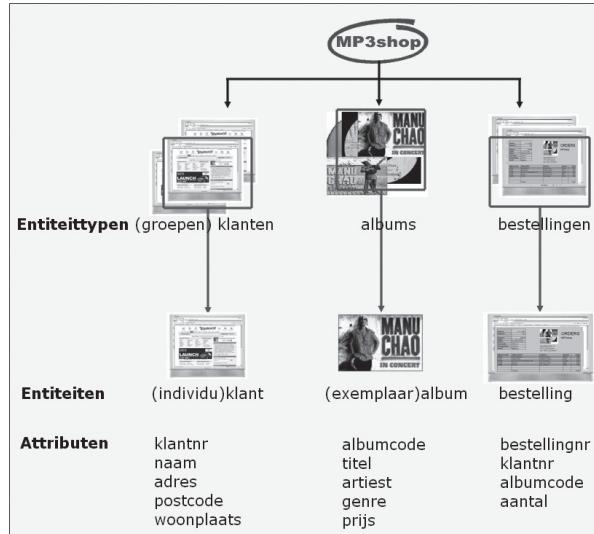
Figuur 5.2
Entiteiten



5.1.6 Wat zijn attributen?

Een attribuut is een *kenmerk* van een entiteit. Bijvoorbeeld naam is een attribuut van de entiteit klant.

Figuur 5.3
Attributen

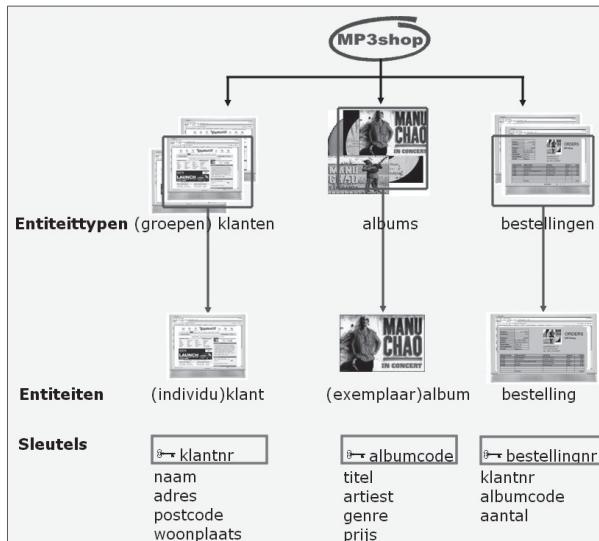


5.1.7 Wat zijn sleutels?

Een sleutel is een attribuut die een entiteit (uniek) identificeert. Bijvoorbeeld albumcode is een unieke identificatie van een album. Dit noemen we een *primary key* (PK) of primaire sleutel.

Figuur 5.4

Sleutels



Opgave 1 Maak Toets 2, hoofdstuk 5

5.2 Normaliseren

5.2.1 Wat is normaliseren?

Normaliseren is een bottom-upaanpak om vanuit bekende informatie een gegevensmodel te ontwerpen. Door het normaliseren van gegevens voorkom je drie problemen:

Figuur 5.5

Onlinebestelling

The screenshot shows a Microsoft Internet Explorer window displaying an online order form for MP3shop. The form includes fields for customer information (Datum: 6-7-2007, Bestellingnr: 101, Klantnr: 20000, Naam: K. Lant Audio, Adres: Einsteinstraat 1, Postcode: 1983 AC, Woonplaats: Arnhem, email: fa@rmv.nl) and a summary table for the order. The summary table has columns: albumcode, titel, artiest, genre, prijs, aantal, and bedrag. It lists several items, including 'Cafe Atlantico' by Cesaria Evora, 'Runba Azul' by Caetano Veloso, 'Der Her ist mein getreuer Hirt' by Ton Koopman, 'Survivor' by Destiny's Child, and 'Oh Girl' by The Chi-lites. The total amount is 19,40. The page header reads 'BESTELLING' and 'MP3shop'.

albumcode	titel	artiest	genre	prijs	aantal	bedrag
A1303	Cafe Atlantico	Cesaria Evora	World	3,00	1	3,00
A1101	Runba Azul	Caetano Veloso	Latin	4,90	1	4,90
A1505	Der Her ist mein getreuer Hirt	Ton Koopman	Klassiek	5,50	1	5,50
A1404	Survivor	Destiny's Child	R&B	3,00	1	3,00
A1606	Oh Girl	The Chi-lites	Other	3,00	1	3,00
						19,40

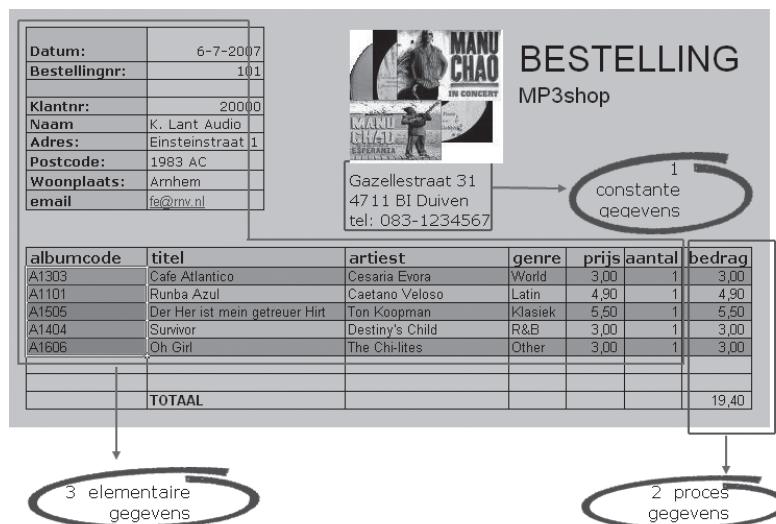
1. Overbodige gegevens: dat zijn gegevens die meermalen ingevoerd moeten worden.
2. Inconsistente gegevens: stel dat het adres van een klant verandert en dat deze wijziging op meerdere plaatsen moet worden aangebracht. Als dat niet gebeurt, heeft dit een inconsistente (tegenstrijdige) gegevensverzameling tot gevolg.
3. Inefficiëntie: het is zeer inefficiënt om dezelfde gegevens op meerdere plaatsen bij te houden.

Dit soort problemen (overbodige, inconsistente en inefficiënte gegevens) kan worden voorkomen met *normaliseren*.

5.2.2 Soorten gegevens

We nemen als voorbeeld de volgende gegevens van een onlinebestelling van de MP3shop. Er zijn drie verschillende soorten gegevens:

Figuur 5.6
Soorten gegevens



1. Constante gegevens: dat zijn gegevens die zelden gewijzigd worden. Bijvoorbeeld naam en adresgegevens van een bedrijf of een btw-percentage.
2. Procesgegevens: procesgegevens kunnen uit andere gegevens afgeleid worden, bijvoorbeeld: bedrag of totaal.
3. Elementaire gegevens: dat zijn alle andere gegevens.

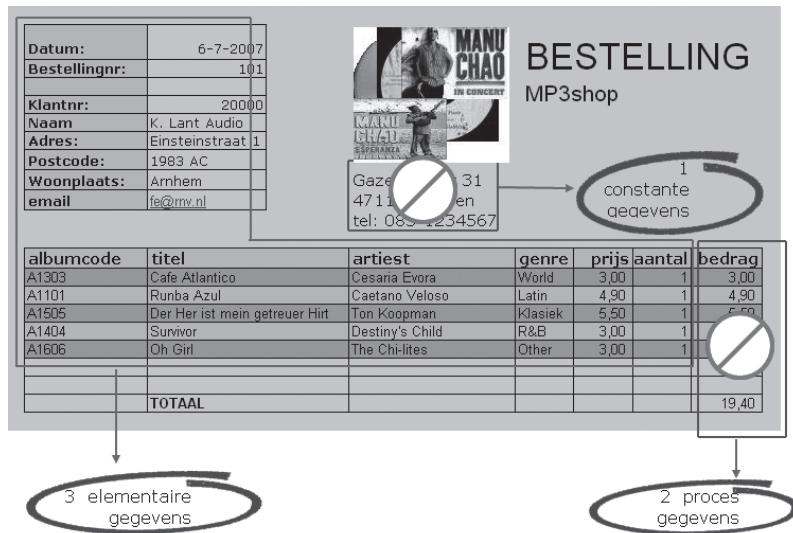
5.3 Normalisatieproces

Het normalisatieproces neemt de volgende vormen:

- De nulde normaalvorm.
- De eerste normaalvorm.
- De tweede normaalvorm.
- De derde normaalvorm.

Figuur 5.7

De nulde
normaalvorm



5.3.1 Nulde normaalvorm (ONV)

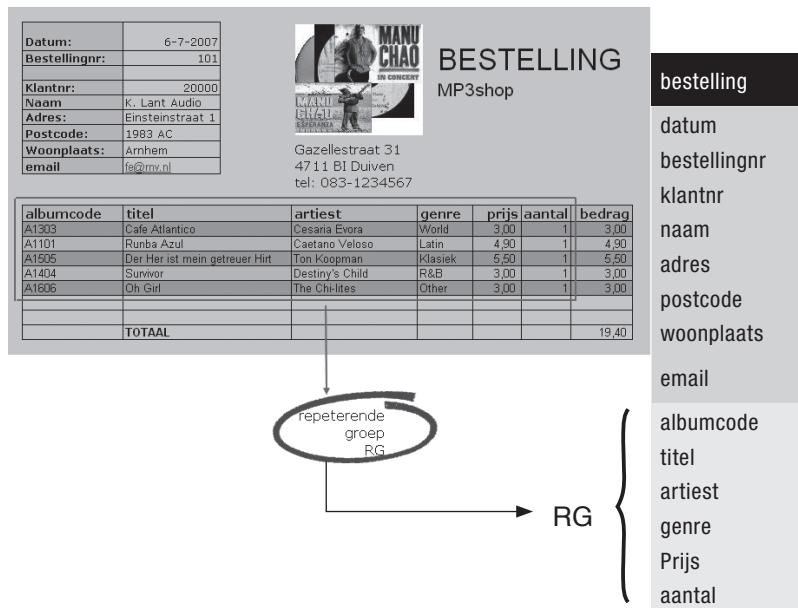
Stap 1: Maak een groep met alleen de elementaire gegevens. Constante gegevens zoals bedrijfsadres en telefoonnummer neem je niet mee. Ook procesgegevens zoals bedragen en totalen neem je niet mee.

bestelling
datum
bestellingnr
klantnr
naam
adres
postcode
woonplaats
email
albumcode
titel
artiest
genre
prijs
aantal

Stap 2: Geef de repeterende groep aan. In een besteling kunnen meerdere albumtitels worden besteld. Dat is de repeterende groep (RG).

Figuur 5.8

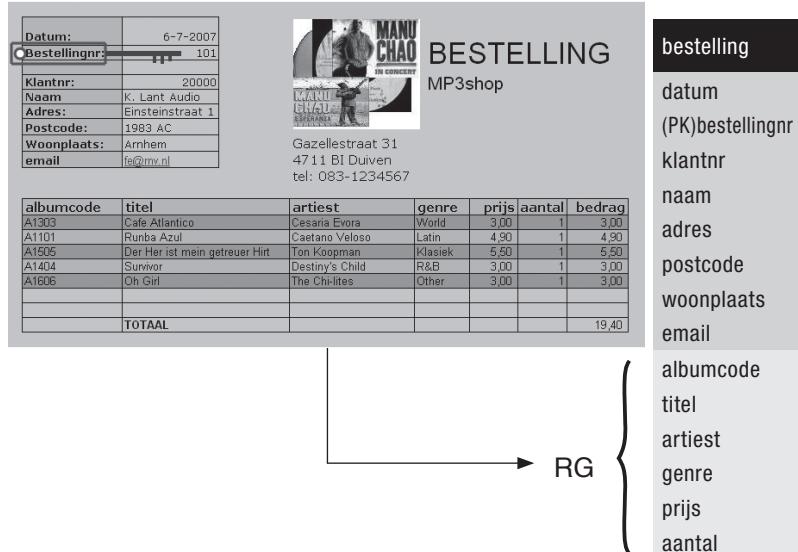
Vind de
repeterende groep



Stap 3: Geef de primaire sleutel aan (PK —). Een primaire sleutel identificeert een unieke entiteit. In dit geval identificeert bestellingnr een unieke bestelling.

Figuur 5.9

Vind en
onderstreep de
primaire sleutel



5.3.2 De eerste normaalvorm (1NV)

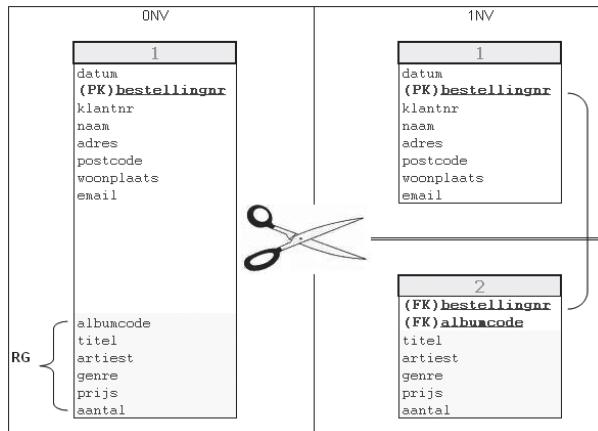
Van de nulde normaalvorm naar de eerste normaalvorm zijn twee stappen.

Stap 1: Maak een tweede groep met de repeterende groep plus een kopie van de primaire sleutel. Deze sleutel wordt een *foreign key* (FK) of verwijzende sleutel en vormt de koppeling tussen de twee groepen.

Stap 2: Bepaal de samengestelde sleutel (twee of meer sleutels) van de tweede groep. Bijvoorbeeld:

Hier maak je een tweede groep met een kopie van bestellingnr plus de repeterende groep (RG). Vervolgens maak je een samengestelde sleutel met bestellingsnr plus albumcode.

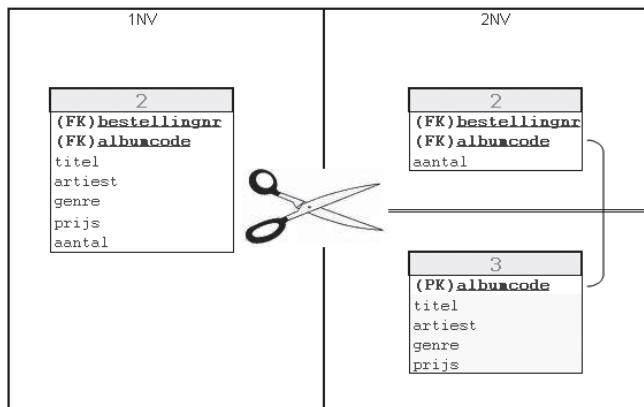
Figuur 5.10
De eerste
normaalvorm



5.3.3 De tweede normaalvorm (2NV)

Maak een derde groep met een kopie van het tweede deel van de samengestelde sleutel in groep 2 plus de bijbehorende attributen. Het tweede deel van de samengestelde sleutel vormt de koppeling tussen de twee groepen. Bijvoorbeeld:

Figuur 5.11
De tweede
normaalvorm



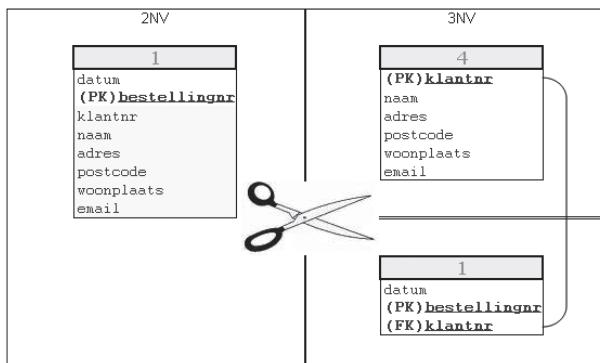
Hier wordt groep drie gemaakt met albumcode en de bijbehorende attributen (alle attributen over album).

5.3.4 De derde normaalvorm (3NV)

Kijk of er een groep is die verder opgesplitst kan worden. Als dat zo is, maak je een vierde groep met de overtollige attributen. Maak een koppeling tussen de twee groepen met een primaire sleutel en een verwijzende sleutel. Bijvoorbeeld:

Figuur 5.12

De derde
normaalvorm

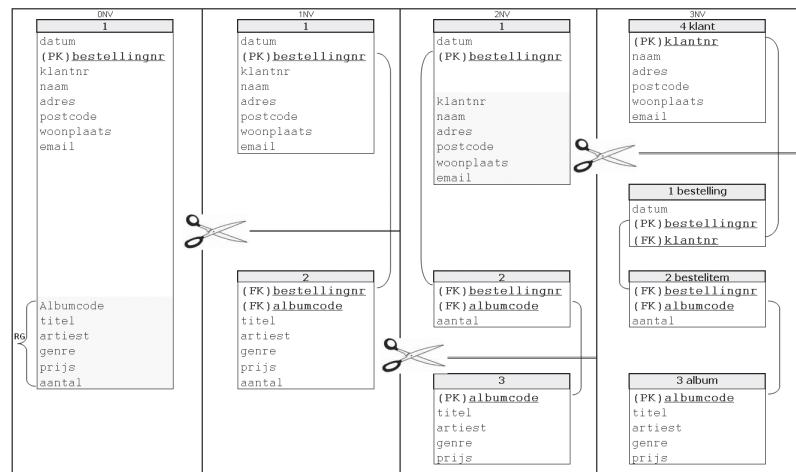


Hier maak je een vierde groep met klantnr en bijbehorende attributen.

5.3.5 Samenvatting

Figuur 5.13

Samenvatting van
het normalisatie-
proces



Bovenstaand proces kun je ook als volgt beschrijven:

bestelling (**bestellingnr**, klantnr, datum)

bestelitem (**bestellingnr**, **albumcode**, aantal)

album (**albumcode**, titel, artiest, genre, prijs)

klant (**klantnr**, naam, adres, postcode, woonplaats, email)

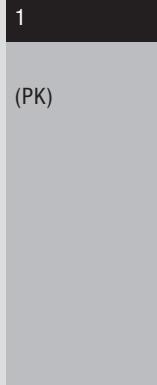
Opgave 2 1. Maak Toets 3, hoofdstuk 5

Figuur 5.14
Onlinefactuur

Datum:	6-7-2007
Bestellingnr:	101
Factuurnr:	10099
Klantnr:	20000
Naam:	K. Lant Audio
Adres:	Einsteinstraat 1
Postcode:	1983 AC
Woonplaats:	Arnhem
email:	fe@rmv.nl

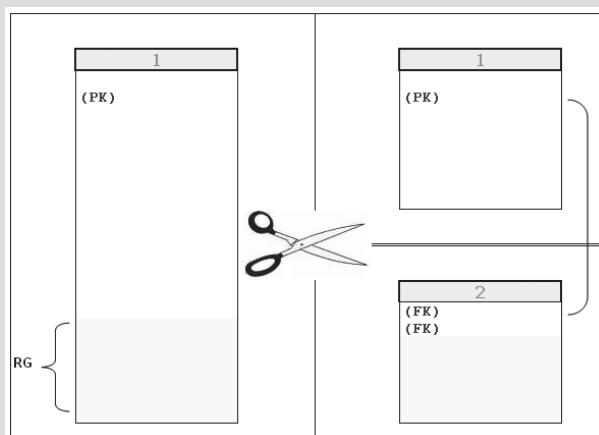
albumcode	titel	artiest	genre	prijs	aantal	bedrag
AU303	Cafe Atlantico	Cesaria Evora	World	3,00	1	3,00
AU101	Runba Azul	Caetano Veloso	Latin	4,90	1	4,90
AU505	Der Her ist mein getreuer Hirt	Ton Koopman	Klassiek	5,50	1	5,50
AU404	Survivor	Destiny's Child	R&B	3,00	1	3,00
AU606	Oh Girl	The Chi-lites	Other	3,00	1	3,00
	Subtotaal					19,40
	Korting 10%					1,94
	Subtotal					17,46
	BTW 19,5%					3,40
	Te Betalen					20,86

2. Maak de nulde normaalvorm van de onlinefactuur uit figuur 5.14.



3. Maak de eerste normaalvorm van de onlinefactuur.

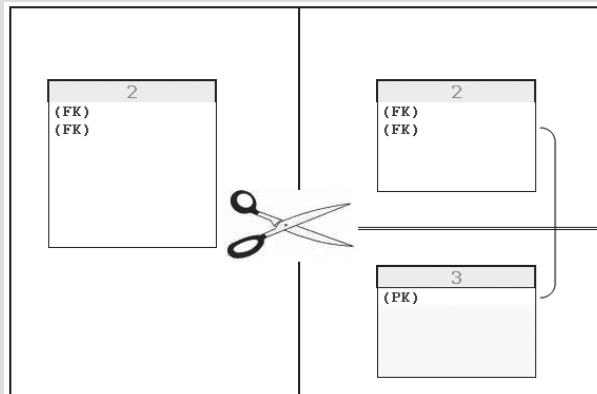
Figuur 5.15
De eerste
normaalvorm
van de online-
factuur



4. Maak de tweede normaalvorm van de onlinefactuur.

Figuur 5.16

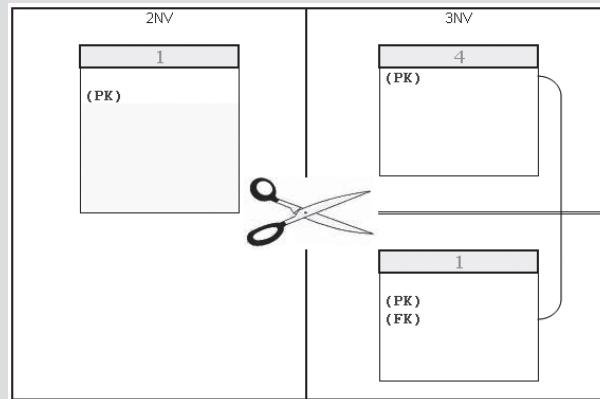
De tweede normaalvorm van de onlinefactuur



5. Maak de derde normaalvorm van de onlinefactuur.

Figuur 5.17

De derde normaalvorm van de onlinefactuur



Database code camp - Lab 1

Normaliseren

De Wijnshop verkoopt kwaliteitswijn aan een klantenbestand met wijnliefhebbers:

- Elk factuur heeft een factuurnummer.
- Elke klant heeft een klantnummer.
- De artikelcode identificeert een uniek wijn.

De Wijnshop heeft momenteel een factureringssysteem dat gebaseerd is op papier. Hieronder zie je een exemplaar van een factuur. De Wijnshop wil een database ontwikkelen voor het factureren van zijn klanten.

Bestudeer onderstaand document en normaliseer de gegevens tot en met de derde normaalvorm.

Figuur 5.18

Een factuur van
De Wijnshop

Factuur					
De Wijnshop					
Klantnr: 0123					
Frans en Ellen Dijkstra					
Servatiusstraat 3					
2385 ZX Zoeterwoude					
Factuurnr 99021					
Besteldatum 02/25/2010					
Code	Artikel	Aantal	Prijs/stuk	Bedrag	
001	Sélection Maitre de Chais; Vaqueyras rouge	2	€ 42,60	€ 85,20	
005	Rosé des Demoiselles; CdR rosé	1	€ 31,90	€ 31,90	
Subtotaal € 117,10					
BTW € 25,25					
Te betalen € 139,35					

Database code camp - Lab 2

Normaliseren

Boekhandel MAXX bestelt boeken van een aantal uitgevers:

- Elke bestelling heeft een bestelnummer.
- Elke uitgever heeft een uniek uitgevernummer.
- ISBN uniek identificeert een titel.

Boekhandel MAXX heeft momenteel een order systeem dat gebaseerd is op papier. Hieronder zie je een exemplaar van een bestelling. Boekhandel MAXX wil een database ontwikkelen voor het bestellen van zijn boeken. Bestudeer onderstaand document en normaliseer de gegevens tot en met de derde normaal vorm.

Figuur 5.19

Een bestelling van
Boekhandel MAXX

BOEKHANDEL MAXX					
Bestelling					
Bestelnummer 91841					Datum: 24 mei 2010
Uitgever :012					
Naam :Educaboek BV					
Adres :Postbus 48					
Woonplaats :4100 AA Culemborg					
ISBN	TITEL	JAAR	SCHRIJVER	AANTAL	
901101684X	Werkboek Lotus 123	1990	Raats e.a.	29	
9011006135	Pascal	1989	Cremers	15	
9011016284	Calc	1990	Mulder	26	
9011008650	Gegevens Analyse	1988	De Boer	18	

5.4 Data modellering

Na de normalisatie van de gegevens gaan we een datamodel ontwerpen. Data modellering is een grafische weergave van de relaties (samenvang) tussen de entiteittypen. Deze relaties geven we schematisch weer in een Entiteit Relatie Diagram (ERD).

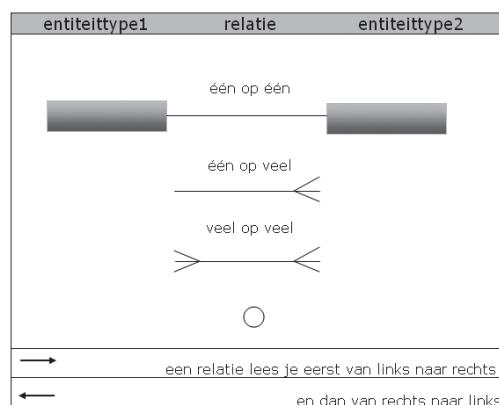
5.4.1 Wat is een ERD?

Een Entiteit Relatie Diagram is een zeer abstracte beschrijving van een datamodel. Een ERD omvat de grafische elementen die je nodig hebt om de relaties tussen de entiteittypen in het datamodel te beschrijven.

Een relatie wordt in twee richtingen gelezen: van links naar rechts en van rechts naar links.

Figuur 5.20

Bij een ERD
gebruiken we de
volgende symbolen

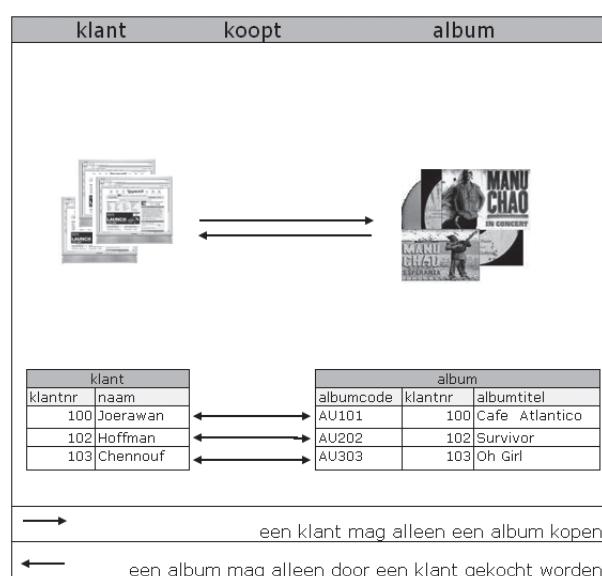


5.4.2 Eén-op-éénrelatie (1:1)

Een één-op-éénrelatie lees je als volgt:

Figuur 5.21

Voorbeeld van
één-op-éénrelatie



- In dit voorbeeld heeft klantnr 100 een album gekocht.
- Albumcode AU303 werd alleen door klantnr 103 gekocht.

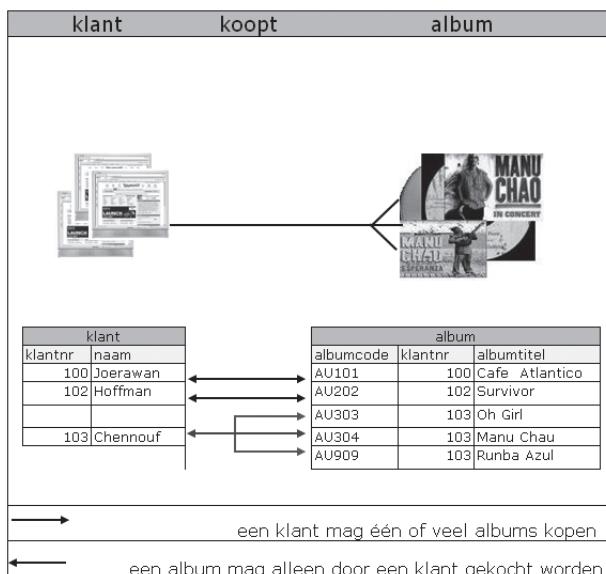
5.4.3 Eén-op-veelrelatie (1:n)

Een één-op-veelrelatie lees je als volgt:

- In dit voorbeeld heeft klantnr 103 drie albums gekocht.
- Albumcode AU303 werd alleen door een klant gekocht.

Figuur 5.22

Voorbeeld van één-op-veelrelatie

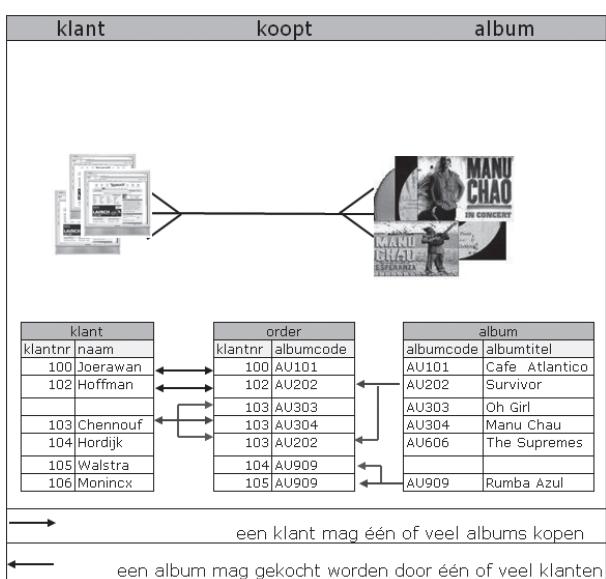


5.4.4 Veel-op-veelrelatie (m:n)

Een veel-op-veelrelatie lees je als volgt:

Figuur 5.23

Voorbeeld van veel-op-veelrelatie



Voor een veel-op-veelrelatie maak je een derde tabel, zoals de ordertabel uit figuur 5.23.

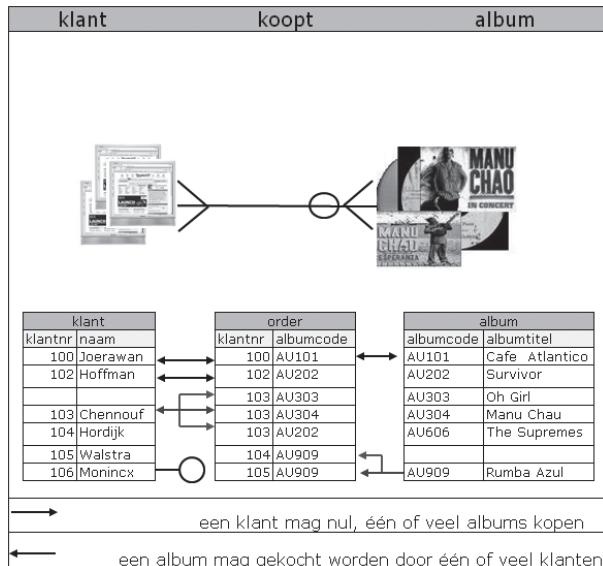
- In dit voorbeeld heeft klantnr 103 drie albums gekocht.
- Albumcode AU909 werd gekocht door twee klanten.

5.4.5 Nul-optionaliteit

Een nul-optionaliteit betekent dat er *niet* per se een relatie hoeft te zijn. Een nul-optionaliteit lees je als volgt:

Figuur 5.24

Voorbeeld van een nul-optionaliteit

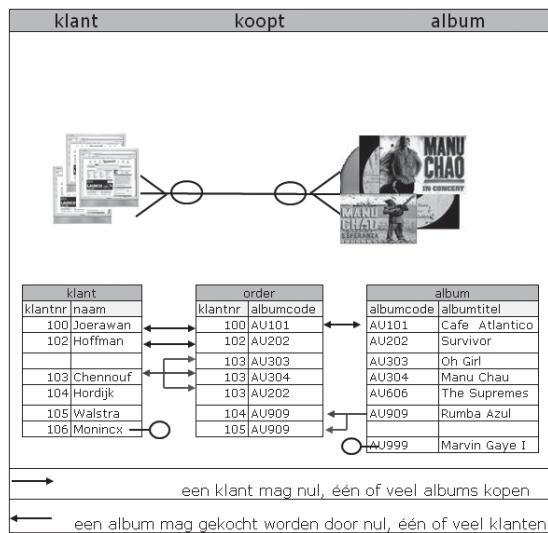


- In dit voorbeeld heeft klantnr 106 nog geen albums gekocht.
- Albumcode AU909 werd gekocht door twee klanten.

5.4.6 Volledige optionaliteit

Een volledige optionaliteit betekent dat er *niet* per se een volledige relatie moet zijn. Een volledige optionaliteit lees je als volgt:

Figuur 5.25
Voorbeeld van een volledige optionaliteit

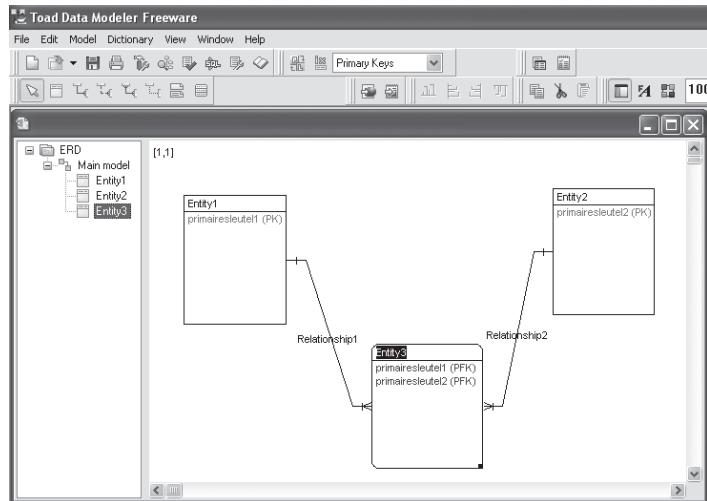


- In dit voorbeeld heeft klantnr 106 nog geen albums gekocht.
- Albumcode AU999 is nog niet verkocht.

5.4.7 Datamodellering-software

Je zou een freeware datamodellering-programma, zoals *Toad Data Modeler* van Quest Software kunnen gebruiken. De homepage van Quest Software vind je op:
http://www.quest.com/Toad_Data_Modeler/.

Figuur 5.26
Voorbeeld van een ERD gemaakt met Toad Data Modeler



Database code camp - Lab 3**<erd>**

Maak een Entiteit Relatie Diagram (ERD) met de relaties tussen de tabellen die je in Lab 1 gemaakt hebt voor De Wijnshop.

Database code camp - Lab 4**<erd>**

Maak nu ook een Entiteit Relatie Diagram (ERD) met de relaties tussen de tabellen die je in Lab 2 gemaakt hebt voor Boekhandel MAXX.

Maak nu Toets 2, hoofdstuk 5 en Toets 3, hoofdstuk 5 en Praktijkopdracht 2, hoofdstuk 5

6

SQL

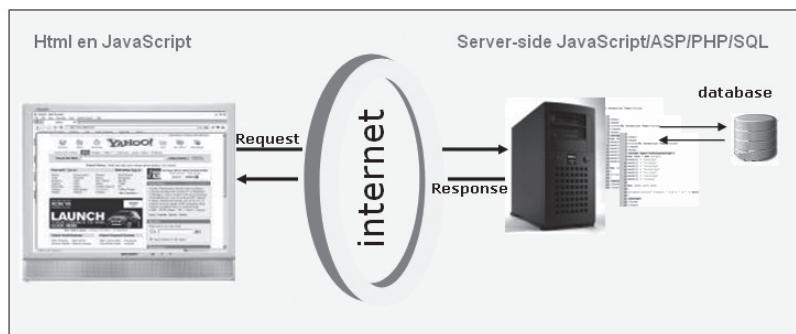
Nadat je een realisatie van de gegevens en een datamodel met tabellen en relaties (ERD) ontworpen hebt, kun je je database met SQL maken.

6.1 Wat is SQL?

SQL staat voor *Structured Query Language*. Dit is een algemene taal die het mogelijk maakt gegevens in een database te zetten of gegevens op te halen. SQL wordt gebruikt om gegevens tussen een applicatie en een database uit te wisselen en te bewerken.

Figuur 6.1

Een webdatabase applicatie



6.2 MySQL

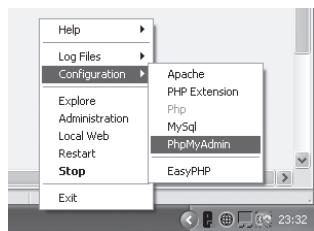
MySQL is een populair databasemanagementsysteem dat in combinatie met PHP wordt gebruikt om volledige en stabiele webdatabaseapplicaties te ontwikkelen. MySQL is een open source relationeel databasemanagementsysteem (RDBMS) dat gebruikmaakt van SQL. De homepage van MySQL vind je op: <http://www.mysql.com>.

6.2.1 phpMyAdmin

Een bekend MySQL-frontend is phpMyAdmin. Als je EasyPHP hebt geïnstalleerd, kun je phpMyAdmin als volgt opstarten:

- Rechtsklikken op EasyPHP. Kies Configuration en vervolgens PhpMyAdmin.

Figuur 6.2
Start PhpMyAdmin



6.2.2 Het maken van een database

In phpMyAdmin maken we een database met de naam **MP3shop**. Geef de naam van de database aan zoals in figuur 6.3 en klik vervolgens op aanmaken. Het volgende scherm verschijnt.

Figuur 6.3
Maak een nieuwe database

6.2.3 Het maken van een tabel

Hier maken we nu een tabel met de naam album met vijf velden:

- Klik op Start, dan zie je het volgende scherm.

Figuur 6.4
Maak een nieuwe tabel

6.2.4 De kolommen benoemen

We benoemen nu de velden en geven de lengte aan zoals in figuur 6.5.

- Klik op Opslaan, dan verschijnt het volgende scherm.

Figuur 6.5

Maak de velden

Veld	Type [Documentatie]	Lengte/Waarden*	Attributen	Null
albumcode	VARCHAR	5		not null
albumtitel	VARCHAR	20		not null
artiest	VARCHAR	20		not null
genre	VARCHAR	20		not null
prijs	DECIMAL	5,2		not null

6.2.5 Hoofdmenu

Om gegevens in te voeren klik je op Invoegen.

- Klik op Invoegen, dan verschijnt het volgende scherm.

Figuur 6.6

Hoofdmenu

Veld	Type	Attributen	Null	Standaardwaarde	Extra	Actie
albumcode	varchar(5)	Nee				
albumtitel	varchar(20)	Nee				
artiest	varchar(20)	Nee				
genre	varchar(20)	Nee				
prijs	decimal(5,2)	Nee	0.00			

6.2.6 Gegevens invoeren

Voer de volgende gegevens in zoals in figuur 6.7.

- Klik op Voeg een nieuw record toe.
- Klik op Start.
- Voer alle albumgegevens in. Klik elke keer op Start.

Figuur 6.7

Gegevens invoeren

Veld	Type	Functie	Null	Waarde
albumcode	varchar(5)		✓	AU303
albumtitel	varchar(20)		✓	Cafe Atlantico
artiest	varchar(20)		✓	Cesarie Evora
genre	varchar(20)		✓	World
prijs	decimal(5,2)		✓	3.00

Voeg toe als nieuwe rij -- En --

Terug
 Voeg een nieuw record toe
 Start Opnieuw

Opgave

- Maak de volgende tabellen in je mp3shop-database.

Figuur 6.8

De tabel Album

albumcode	titel	artiest	genre	prijs
A1303	Cafe Atlantico	Cesarie Evora	World	3.00
A1101	Rumba Azul	Caetano Veloso	Latin	4.90
A1404	Survivor	Destiny's Child	R&B	3.00
A1606	Oh Girl	The Chi-lites	Pop	3.00
A1505	Der Her ist mei getreu	Ton Koopman	Klasiek	5.50
A1999	Closing Time	Tom Waits	Rock	3.00
A1123	Irresistible	Celia Cruz	Latin	3.50
A1124	Marvin Gaye II	Marvin Gaye	R&B	4.00
A1321	Mi Sangre	Juanes	Latin	3.90
A2001	Greatest Hits 2	Queen	Rock	3.00
A2003	3121	Prince	Rock	3.45
A2444	Antologia I	Paco de Lucia	World	3.00

Figuur 6.9
De tabel Klant

klantnr	naam	adres	postcode	woonplaats	email
K1101	Dylan Huisden	Middenweg 11	1088VV	Amsterdam	dhuisden@roc.nl
K1102	Nitin Bosman	Leidseweg 22	9900BB	Amsterdam	nbosman@roc.nl
K1103	Joseph Demirel	Leidseplein 33	9988BB	Utrecht	josdem@hotmail.com
K1000	Franco Tasiyan	Kruislaan 444	3300VV	Utrecht	frantas@wanadoo.nl
K1901	Akash Kabli	Biohof 55	2299NN	Amstelveen	aka@hetnet.nl
K1902	Tamara Kabli	Mozartstraat 22	3388XX	Amsterdam	tamka@hotmail.com
K1100	Arnold Shaw	Kruislaan 1	9876FF	Rotterdam	asha@roc.nl

Figuur 6.10
De tabel Bestelling

bestellingnr	klantnr	datum
B1201	K1101	2007-01-01
B1203	K1102	2007-01-01
B1204	K1103	2007-02-15
B1205	K1000	2007-02-20
B1206	K1902	2007-03-13

Figuur 6.11
De tabel
Bestelitem

bestellingnr	albumcode	aantal
B1201	A1303	1
B1201	A1101	1
B1201	A1444	1
B1203	A1101	1
B1203	A1444	1
B1204	A1123	1
B1204	A1124	1
B1204	A2001	1
B1205	A1444	1
B1205	A1101	1
B1205	A1003	1
B1206	A1101	1
B1206	A1123	1
B1206	A1124	1
B1206	A1303	1
B1206	A1505	1

6.3 De databaseadministrator

De databaseadministrator (DBA) is verantwoordelijk voor het ontwerpen, implementeren en onderhouden van de databases binnen een organisatie. De rol van de DBA is het monitoren en verbeteren van de database performance en veiligheid.

MySQL heeft een interface phpMyAdmin voor de databaseadministrator en voor de webdeveloper. Start de databaseserver op en open phpMyAdmin. Open de mp3shop-database gemaakt in paragraaf 6.2 zoals hieronder:

Figuur 6.12

De interface van phpMyAdmin

De mp3shop databasepagina met vier tabellen verschijnt. Vanuit het hoofdmenu kunnen we de structuur van de tabellen bekijken. We kunnen ook een SQL-venster openen om SQL-opdrachten uit te voeren. Ook het importeren en exporteren van databases kunnen we vanuit het hoofdmenu uitvoeren.

Figuur 6.13

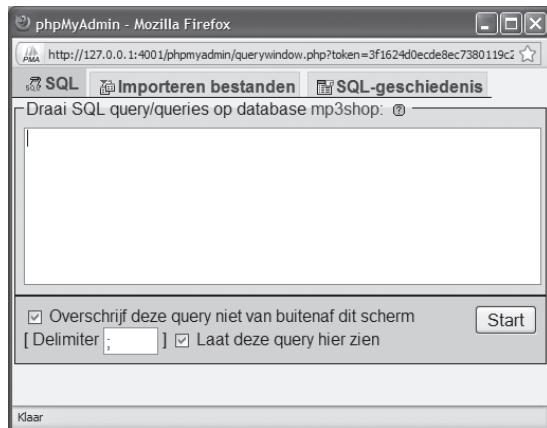
Het hoofdmenu van phpMyAdmin

Tabel	Actie	Records	Type	Collate	Grootte	Overhead
album	X	12	MyISAM	latin1_general_ci	2, 5 KB	-
bestelitem	X	16	MyISAM	latin1_general_ci	1, 3 KB	-
bestelling	X	5	MyISAM	latin1_general_ci	2, 1 KB	-
klant	X	7	MyISAM	latin1_general_ci	2, 6 KB	88 Bytes
4 tabel(len)	Som	40	MyISAM	latin1_general_ci	8, 5 KB	88 Bytes

Klik op de SQL-knop en het SQL-venster verschijnt zoals hieronder:

Figuur 6.14

Het SQLvenster.



Vanuit dit venster kunnen we SQL-opdrachten die gebaseerd zijn op de gekozen database uitvoeren. We beginnen met de SELECT-opdracht.

6.3.1 De opdracht SELECT *

De opdracht `SELECT * FROM tabel_naam;` geeft als resultaat een gegevensverzameling van alle kolommen van de gegeven tabel. Syntaxis:

```
SELECT * FROM tabel_naam;
```

- Opgaven** 2 De SQL-codes van alle **Opgaven** sla je in een tekstbestand op. Aan het einde lever je het bestand bij je docent in. Open het SQL-venster, typ de volgende MySQL-opdracht in en bekijk de resultaatverzameling:

```
SELECT * FROM album;
```

- 3 De SQL-codes van alle **Opgaven** sla je in een tekstbestand op. Aan het einde lever je het bestand bij de docent in. Open het SQL-venster en voer een query uit die de tabel **bestelitem** weergeeft. De resultaatverzameling moet er uitzien als in figuur 6.15. Deze verzameling kun je ook afdrukken door op de printknop te drukken.

Figuur 6.15
De resultaatverzameling

Query results operations			
Printopmaak Print overzicht (met volledige teksten)			
Toon: <input type="text" value="30"/> rijen beginnend bij in <input type="text" value="horizontaal"/> modus en herhaal			
← T →	bestellingnr	albumcode	aantal
<input type="checkbox"/>	B1201	A1303	1
<input type="checkbox"/>	B1201	A1101	1
<input type="checkbox"/>	B1201	A1444	1
<input type="checkbox"/>	B1203	A1101	1
<input type="checkbox"/>	B1203	A1444	1
<input type="checkbox"/>	B1204	A1123	1
<input type="checkbox"/>	B1204	A1124	1
<input type="checkbox"/>	B1204	A2001	1
<input type="checkbox"/>	B1205	A1444	1
<input type="checkbox"/>	B1205	A1101	1
<input type="checkbox"/>	B1205	A1003	1
<input type="checkbox"/>	B1206	A1101	1
<input type="checkbox"/>	B1206	A1123	1
<input type="checkbox"/>	B1206	A1124	1
<input type="checkbox"/>	B1206	A1303	1
<input type="checkbox"/>	B1206	A1505	1

6.3.2 SELECT-kolommen

De **SELECT**-opdracht is een zoekopdracht naar specifieke kolommen in een tabel. Syntaxis:

```
SELECT kolom_naam [, kolom_naam] FROM tabel_naam
```

- Opgaven** 4 Open het SQL-venster en voer de volgende query uit en bekijk de resultaatverzameling:

```
SELECT artiest,titel,genre FROM album;
```

- 5 Open het SQL-venster en maak de volgende query: "Selecteer de kolommen **naam** en **adres** vanuit de klanttabel."

De resultaatverzameling moet er zo uitzien:

Figuur 6.16
De kolommen naam en adres geselecteerd

← T →	naam	adres
<input type="checkbox"/>	Dylan Huisden	Middenweg 11
<input type="checkbox"/>	Nitin Bosman	Leidseweg 22
<input type="checkbox"/>	Joseph Demirel	Leidseplein 33
<input type="checkbox"/>	Franco Tasiyan	Kruislaan 444
<input type="checkbox"/>	Akash Kabli	Biohof 55
<input type="checkbox"/>	Tamara Kabli	Mozartstraat 22
<input type="checkbox"/>	Arnold Shaw	Kruislaan 1

6.3.3 ORDER BY

De **ORDER BY**-clausule geeft een gesorteerde resultaatverzameling weer. Syntaxis:

```
SELECT kolom_naam [, kolom_naam] FROM tabel_naam ORDER BY kolom_naam
```

Opgaven

- 6 Open het SQL-venster en voer de volgende query uit en bekijk de resultaatverzameling:

```
SELECT artiest,titel,genre FROM album ORDER BY artiest;
```

- 7 Open het SQL-venster en maak de volgende query: "Selecteer alle kolommen vanuit de albumtabel in titelvolgorde."
De resultaatverzameling moet er zo uitzien:

Figuur 6.17

De resultaatverzameling

<input type="button" value="←"/>	<input type="button" value="→"/>	albumcode	titel	artiest	genre	prijs
<input type="checkbox"/>	<input type="button" value="edit"/>	X A2003	3121	Prince	Rock	3.45
<input type="checkbox"/>	<input type="button" value="edit"/>	X A2444	Antologia I	Paco de Lucia	World	3.00
<input type="checkbox"/>	<input type="button" value="edit"/>	X A1303	Cafe Atlantico	Cesarie Evora	World	3.00
<input type="checkbox"/>	<input type="button" value="edit"/>	X A1999	Closing Time	Tom Waits	Rock	3.00
<input type="checkbox"/>	<input type="button" value="edit"/>	X A1505	Der Her ist mei getreu	Ton Koopman	Klasiek	5.50
<input type="checkbox"/>	<input type="button" value="edit"/>	X A2001	Greatest Hits 2	Queen	Rock	3.00
<input type="checkbox"/>	<input type="button" value="edit"/>	X A1123	Irresistible	Celia Cruz	Latin	3.50
<input type="checkbox"/>	<input type="button" value="edit"/>	X A1124	Marvin Gaye II	Marvin Gaye	R&B	4.00
<input type="checkbox"/>	<input type="button" value="edit"/>	X A1321	Mi Sangre	Juanes	Latin	3.90
<input type="checkbox"/>	<input type="button" value="edit"/>	X A1606	Oh Girl	The Chi-lites	Pop	3.00
<input type="checkbox"/>	<input type="button" value="edit"/>	X A1101	Rumba Azul	Caetano Veloso	Latin	4.90
<input type="checkbox"/>	<input type="button" value="edit"/>	X A1404	Survivor	Destiny's Child	R&B	3.00

6.3.4 WHERE

Met de WHERE-clausule geef je een selectiecriterium aan. Syntaxis:

```
SELECT kolomnamen FROM tabel  
WHERE voorwaarde
```

Bij de WHERE-clausule kunnen we de volgende operatoren met de voorwaarde gebruiken:

Operator	Beschrijving
=	Gelijk aan
<>	Niet gelijk aan
>	Groter dan
<	Kleiner dan
>=	Groter dan of gelijk aan
<=	Kleiner dan of gelijk aan
BETWEEN	Binnen een range
LIKE	Zoek een patroon

- Opgaven 8** Open het SQL-venster en voer de volgende query uit en bekijk de resultaatverzameling:

```
SELECT * FROM klant
WHERE woonplaats= "Amsterdam" ;
```

- 9** Open het SQL-venster en maak de volgende query: “Selecteer alle albums vanuit de albumtabel waarvan het genre ‘Latin’ is.”
De resultaatverzameling moet er zo uitzien:

Figuur 6.18

Resultaatverzameling van het genre Latin

	← ↑ →	albumcode	titel	artiest	genre	prijs
<input type="checkbox"/>		X	A1101	Rumba Azul	Caetano Veloso	Latin 4.90
<input type="checkbox"/>		X	A1123	Irresistible	Celia Cruz	Latin 3.50
<input type="checkbox"/>		X	A1321	Mi Sangre	Juanes	Latin 3.90

6.3.5 BETWEEN ... AND

De BETWEEN ... AND-clausule selecteert een range tussen twee waarden in. De waarden kunnen nummers, tekst of datum zijn. Syntaxis:

```
SELECT kolomnamen FROM tabel
WHERE kolomnaam BETWEEN lagewaarde AND hogewaarde;
```

- Opgaven 10** Open het SQL-venster en voer de volgende query uit en bekijk de resultaatverzameling:

```
SELECT * FROM album
WHERE prijs BETWEEN 3.00 AND 4.00;
```

- 11** Open het SQL-venster en maak de volgende query: “Selecteer alle bestellingen vanuit de bestellingtabel waarvoor de datum tussen ‘2007-01-01’ en ‘2007-02-01’ ligt.”
De resultaatverzameling moet er zo uitzien:

Figuur 6.19

De resultaatverzameling voor de datumrange

	← ↑ →	bestellingnr	klantnr	datum
<input type="checkbox"/>		X	B1201	K1101 2007-01-01
<input type="checkbox"/>		X	B1203	K1102 2007-01-01

6.3.6 LIMIT

De LIMIT-clausule geeft het aantal rijen uit je resultaatverzameling dat je wilt zien. Syntaxis:

```
SELECT kolomnamen FROM tabel LIMIT integer;
```

- Opgaven 12** Open het SQL-venster en voer de volgende query uit en bekijk de resultaatverzameling:

```
SELECT * FROM album
WHERE prijs BETWEEN 3.00 AND 4.00 LIMIT 5;
```

- 13** Open het SQL-venster en maak de volgende query: “Selecteer de eerste drie bestellingen vanuit de bestellingtabel.”

De resultaatverzameling moet er zo uitzien:

Figuur 6.20

De resultaatverzameling van de eerste drie bestellingen

	bestellingnr	klantnr	datum
<input type="checkbox"/>	B1201	K1101	2007-01-01
<input type="checkbox"/>	B1203	K1102	2007-01-01
<input type="checkbox"/>	B1204	K1103	2007-02-15

6.3.7 SELECT DISTINCT

De DISTINCT-clausule geeft een resultaatverzameling zonder dubbele gegevens. Bijvoorbeeld in de bestelitemtabel heb je meerdere items met hetzelfde bestellingnr. Syntaxis:

```
SELECT DISTINCT colomnamen FROM tabelnaam;
```

Opgaven 14 Open het SQL-venster en voer de volgende query uit en bekijk de resultaatverzameling:

```
SELECT DISTINCT bestellingnr FROM bestelitem;
```

15 Open het SQL-venster en maak de volgende query: “Selecteer distinct albumcode uit de bestelitemtabel.”

De resultaatverzameling moet er zo uitzien:

Figuur 6.21

De resultaatverzameling van de albumcode

	albumcode
<input type="checkbox"/>	A1303
<input type="checkbox"/>	A1101
<input type="checkbox"/>	A1444
<input type="checkbox"/>	A1123
<input type="checkbox"/>	A1124
<input type="checkbox"/>	A2001
<input type="checkbox"/>	A1003
<input type="checkbox"/>	A1505

6.3.8 SELECT WHERE LIKE

De WHERE LIKE-clausule geeft een resultaatverzameling gebaseerd op de vergelijking met een patroon. In het patroon kunnen we het wildcard teken (%) gebruiken. Bijvoorbeeld in de albumtabel kunnen we een selectie vragen naar alle albums waar het woord ‘azul’ in de titel voorkomt.

Syntaxis:

```
SELECT * FROM tabelnaam WHERE kolomnaam LIKE patroon;
```

Opgaven 16 Open het SQL-venster en voer de volgende query uit en bekijk de resultaatverzameling:

```
SELECT * FROM album WHERE title LIKE '%azul%';
```

- 17** Open het SQL-venster en maak de volgende query: “Selecteer alle albums waar het woord ‘Girl’ in de titel voorkomt.”
De resultaatverzameling moet er zo uitzien:

Figuur 6.22

De resultaatverzameling voor het woord Girl in de titel

albumcode	titel	artiest	genre	prijs	voorraad
A1606	Oh Girl	The Chi-lites	Pop	3.00	2

6.3.9 INNER JOIN

We hebben een datamodel ontworpen waarin iedere tabel een primaire of verwijzende sleutel heeft. Zo kunnen we een relatie tussen de tabellen maken. Soms moeten we gegevens vanuit twee of meer tabellen selecteren. In deze gevallen gebruiken we een `INNER JOIN`-opdracht. Syntaxis:

```
SELECT kolom1, kolom2, kolom3
FROM eerste _ tabel
INNER JOIN tweede _ tabel
ON eerste _ tabel.primaire _ sleutel = tweede _ tabel.
verwijzende _ sleutel
```

- Opgaven 18** Open het SQL-venster en maak de volgende query: “Wie heeft welke bestellingen geplaatst?”
Bekijk daarna de resultaatverzameling.

- 19** Open het SQL-venster en voer de volgende opdracht uit: Maak een `SELECT JOIN`-opdracht die de volgende vraag beantwoordt: Hoeveel albums Rumba Azul zijn besteld?”
De resultaatverzameling moet er zo uitzien:

Figuur 6.23

De resultaatverzameling van het aantal albums Rumba Azul

← T →		
artiest	albumcode	aantal
Caetano Veloso	A1101	1

6.3.10 LEFT JOIN

De `LEFT JOIN`-clausule geeft alle rijen weer vanuit de eerste tabel, zelfs als er geen link is naar de tweede tabel. Syntaxis:

```
SELECT kolom1, kolom2, kolom3
FROM eerste _ tabel
LEFT JOIN tweede _ tabel
ON eerste _ tabel.primaire _ sleutel = tweede _ tabel.
verwijzende _ sleutel;
```

Opgave 20 Open het SQL-venster en maak de volgende query: "Wie heeft wel en wie geen bestellingen geplaatst?"

De resultaatverzameling moet er zo uitzien:

Figuur 6.24

De resultaatverzameling van de bestellers. Je ziet dat Akash en Arnold nog niet hebben besteld

naam	email	bestellingnr	datum
Dylan Huisden	dhuisden@roc.nl	B1201	2007-01-01
Nitin Bosman	nbosman@roc.nl	B1203	2007-01-01
Joseph Demirel	josdem@hotmail.com	B1204	2007-02-15
Franco Tasiyan	frantas@wanadoo.nl	B1205	2007-02-20
Akash Kabli	aka@hetnet.nl	NULL	NULL
Tamara Kabli	tamka@hotmail.com	B1206	2007-03-13
Arnold Shaw	asha@roc.nl	NULL	NULL

21 Open het SQL-venster en voer de volgende opdracht uit: "Maak een SELECT album LEFT JOIN bestelitem-opdracht die de volgende vraag beantwoordt: welke albums zijn besteld en welke albums niet?"

De resultaatverzameling moet er zo uitzien:

Figuur 6.25

De resultaatverzameling van welke albums wel en niet besteld zijn

titel	bestellingnr	aantal
Cafe Atlantico	B1201	1
Cafe Atlantico	B1206	1
Rumba Azul	B1201	1
Rumba Azul	B1203	1
Rumba Azul	B1205	1
Rumba Azul	B1206	1
Survivor	NULL	NULL
Oh Girl	NULL	NULL
Der Her ist mei getreu	B1206	1
Closing Time	NULL	NULL
Irresistible	B1204	1
Irresistible	B1206	1
Marvin Gaye II	B1204	1
Marvin Gaye II	B1206	1
Mi Sangre	NULL	NULL
Greatest Hits 2	B1204	1
3121	NULL	NULL
Antologia I	NULL	NULL

6.3.11 INNER JOIN

Stel je wilt vier tabellen in je database raadplegen met de volgende vraag: "Welke klanten hebben het album Rumba Azul besteld?" Dan moet je de tabellen Klant, Bestelling, Bestelitem en Album raadplegen met drie INNER JOIN SELECT-opdrachten.

Opgave 22 Open het SQL-venster en maak de volgende query:

```
SELECT klant.naam, klant.email, album.titel,
       album.artiest, bestelitem.bestellingnr, bestelitem.aantal
  FROM klant
    INNER JOIN (bestelling
    INNER JOIN (bestelitem
    INNER JOIN album
```

```

ON album.albumcode = bestelitem.albumcode)
ON bestelling.bestellingnr = bestelitem.bestellingnr)
ON klant.klantnr = bestelling.klantnr
WHERE bestelitem.albumcode = 'A1101';

```

De resultaatverzameling moet er zo uitzien:

Figuur 6.26

De resultaatverzameling

naam	email	titel	artiest	bestellingnr	aantal
Dylan Huisden	dhuisden@roc.nl	Rumba Azul	Caetano Veloso	B1201	1
Nitin Bosman	nbosman@roc.nl	Rumba Azul	Caetano Veloso	B1203	1
Franco Tasiyan	frantas@wanadoo.nl	Rumba Azul	Caetano Veloso	B1205	1
Tamara Kabli	tamka@hotmail.com	Rumba Azul	Caetano Veloso	B1206	1

6.3.12 COUNT

De COUNT-clausule geeft als resultaat een optelling van de gegeven kolom.
Syntaxis:

```
SELECT kolomnamen, COUNT(kolomnaam) FROM tabelnaam GROUP BY kolomnaam;
```

- Opgaven 23** Open het SQL-venster en maak de volgende query: “Wat is het aantal bestellingen per album?”

De resultaatverzameling moet er zo uitzien:

Figuur 6.27

De resultaatverzameling van het aantal bestellingen per album

albumcode	COUNT(aantal)
A1003	1
A1101	4
A1123	2
A1124	2
A1303	2
A1444	3
A1505	1
A2001	1

- 24** Open het SQL-venster en maak de volgende query: “Maak een SELECT COUNT-opdracht die de aantal bestellingen per bestellingnummer optelt.”

De resultaatverzameling moet er zo uitzien:

Figuur 6.28

De resultaatverzameling van het aantal bestellingen per bestellingnummer

bestellingnr	COUNT(aantal)
B1201	3
B1203	2
B1204	3
B1205	3
B1206	5

6.3.13 MIN

De MIN-clausule geeft als resultaat de laagste waarde in de gegeven kolom.
Syntaxis:

```
SELECT MIN(kolomnaam) FROM tabelnaam;
```

- Opgave 25** Open het SQL-venster en maak de volgende query: “Wat is de laagste prijs van een album?”
De resultaatverzameling moet er zo uitzien:

Figuur 6.29

De resultaatverzameling van de laagste prijs per album

← T →
MIN(prijs)
3.00

- Opgave 26** Open het SQL-venster en maak de volgende query: “Wat is de hoogste prijs van een album?”
De resultaatverzameling moet er zo uitzien:

Figuur 6.30

De resultaatverzameling van de hoogste prijs per album

← T →
MAX(prijs)
5.50

- Opgave 27** Open het SQL-venster en maak de volgende query: “Wat is het totale aantal bestelde albums?”
De resultaatverzameling moet er zo uitzien:

Figuur 6.31

De resultaatverzameling van het totaal aantal bestelde albums

← T →
SUM(aantal)
16

6.3.16 UPDATE

We gebruiken de UPDATE-opdracht om gegevens in een tabel te wijzigen.
Syntaxis:

```
UPDATE tabel_naam
SET kolom_naam = nieuwe_waarde
WHERE voorwaarde;
```

Opgave 28 Open het SQL-venster en voer de volgende query uit:

```
UPDATE klant  
SET  
adres = 'Galileiplantsoen 333',  
postcode = '1010RR'  
WHERE klantnr = 'K1000';
```

6.3.17 INSERT INTO

Je weet nu hoe je een tabel aanmaakt, maar natuurlijk wil je ook gegevens aan tabellen kunnen toevoegen. Met de `INSERT INTO`-opdracht kun je nieuwe rijen (records) aan een tabel toevoegen. Syntaxis:

```
INSERT INTO naam_tabel  
VALUES (waarde1, waarde2,...);
```

Je kunt ook kolomnamen gebruiken:

```
INSERT INTO naam_tabel  
(kolom1, kolom2, kolom3...)  
VALUES (waarde1, waarde2, waarde3...);
```

Opgave 29 Open het SQL-venster en voer de volgende `INSERT INTO`-opdracht uit:

```
INSERT INTO klant  
(klantnr,naam,adres,postcode,woonplaats,email)  
VALUES ('K9999','Lex Camilla', 'Hagabakka 24',  
'4656RR','Utrecht', 'lecam@wanadoo.nl')
```

6.3.18 DELETE

De `DELETE`-opdracht verwijdert rijen uit een tabel. Syntaxis:

```
DELETE FROM tabel_naam  
WHERE voorwaarde;
```

Opgave 30 Open het SQL-venster en voer de volgende query uit:

```
DELETE FROM klant  
WHERE klantnr = 'K9999' ";
```

6.3.19 CREATE DATABASE

De `CREATE DATABASE`-opdracht creëert een nieuwe database. Syntaxis:

```
CREATE DATABASE databasenaam;
```

Opgave 31 Open het SQL-venster en voer de volgende query uit:

```
CREATE DATABASE test;
```

6.3.20 DROP DATABASE

De `DROP DATABASE`-opdracht verwijdert een bestaande database.
Syntaxis:

```
DROP DATABASE databasenaam;
```

Opgaven 32 Open het SQL-venster en voer de volgende query uit:

```
DROP DATABASE test;
```

33 Open het SQL-venster en voer de volgende opdracht uit: Maak een nieuwe database met de naam **administratie**.

6.3.21 CREATE TABLE

De `CREATE TABLE`-opdracht creëert een nieuwe tabel. Syntaxis:

```
CREATE TABLE
tabelnaam (
kolomnaam datatype [,kolomnaam datatype]
[,primary key(kolomnaam)]
[,foreign key(kolomnaam)]);
```

Opgave 34 Open het SQL-venster en voer de volgende query uit:

```
CREATE TABLE testtable (naam VARCHAR (15), leeftijd INT(2));
```

6.3.22 DROP TABLE

De `DROP TABLE`-opdracht verwijdert een bestaande tabel. Syntaxis:

```
DROP TABLE tabelnaam;
```

Opgaven 35 Open het SQL-venster en voer de volgende query uit:

```
DROP TABLE testtable;
```

36 Open het SQL-venster en voer de volgende opdracht uit: Maak een nieuwe tabel met de naam **leerlingen** en met de volgende kolommen:

```
Leerlingnummer INT(4),
Achternaam varchar(15),
Voornaam varchar(15),
Leeftijd INT(2),
Adres varchar(15),
Postcode varchar(6),
Woonplaats varchar(15),
Telefoonnummer varchar(10),
Email varchar(15),
Opleiding varchar(15)
```

De resultaatverzameling moet er zo uitzien:

Figuur 6.32

De resultaatverzameling voor de tabel leerlingen

Resultaatverzameling						
	Veld	Type	Collatie	Attributen	Null	Standaardwaarde
<input type="checkbox"/>	Leerlingnummer	int(4)			Ja	NULL
<input type="checkbox"/>	Achternaam	varchar(15)	utf8_general_ci		Ja	NULL
<input type="checkbox"/>	Voornaam	varchar(15)	utf8_general_ci		Ja	NULL
<input type="checkbox"/>	Leeftijd	int(2)			Ja	NULL
<input type="checkbox"/>	Adres	varchar(15)	utf8_general_ci		Ja	NULL
<input type="checkbox"/>	Postcode	varchar(6)	utf8_general_ci		Ja	NULL
<input type="checkbox"/>	Woonplaats	varchar(15)	utf8_general_ci		Ja	NULL
<input type="checkbox"/>	Telefoonnummer	varchar(10)	utf8_general_ci		Ja	NULL
<input type="checkbox"/>	Email	varchar(15)	utf8_general_ci		Ja	NULL
<input type="checkbox"/>	Opleiding	varchar(15)	utf8_general_ci		Ja	NULL

6.3.23 INSERT INTO

De `INSERT INTO`-opdracht voegt een rij of record aan een bestaande tabel toe. Syntaxis:

```
INSERT INTO tabelnaam VALUES(waarde1[,waarde2]);
```

Opgaven 37 Open het SQL-venster en voer de volgende query uit:

```
INSERT INTO leerlingen VALUES(
1002, "Huisden", "Dylan", 18, "Middenweg 11",
"1008VV", "Amsterdam", "0204445555",
"dhuisden@rocva.nl", "ict");
```

38 Open het SQL-venster en voer de volgende opdracht uit: Voeg de volgende student aan de leerlingentabel toe:

```
1003
Bosman
Nitin
17
Leidseweg 22
9900BB
Amstelveen
0209994444
nbosman@hotmail.com
ict
```

De resultaatverzameling moet er zo uitzien:

Resultaatverzameling												
Query results operations												
Printopmaak Print overzicht (met volledige teksten) Exporteer												
Toon:	30	rijen beginnend bij	0									
in	horizontaal	modus en herhaal kopregels na	100	cellen								
<input type="button" value="←"/>	<input type="button" value="→"/>	Leerlingnummer	Achternaam	Voornaam	Leeftijd	Adres	Postcode	Woonplaats	Telefoonnummer	Email	Opleiding	
<input type="checkbox"/>	<input checked="" type="checkbox"/>	1002	Huisden	Dylan	18	Middenweg 11	1008VV	Amsterdam	0204445555	dhuisden@rocva.	ict	
<input type="checkbox"/>	<input checked="" type="checkbox"/>	1003	Bosman	Nitin	17	Leidseweg 22	9900BB	Amstelveen	0209994444	nbosman@hotmail.com	ict	

Figuur 6.33 De resultaatverzameling na het toevoegen van Nitin Bosman

6.3.24 ALTER TABLE

De `ALTER TABLE`-opdracht wijzigt de structuur van een bestaande tabel. Hieronder zie je de meest belangrijk `ALTER TABLE`-opdrachten:

```
RENAME  
MODIFY COLUMN  
ADD COLUMN  
DROP COLUMN  
CHANGE COLUMN  
ADD PRIMARY KEY  
DROP PRIMARY KEY
```

In de volgende paragrafen gaan we deze opdrachten nader bekijken.

6.3.25 RENAME

Met de `RENAME`-opdracht kunnen we de naam van een bestaande tabel wijzigen. Syntaxis:

```
ALTER TABLE tabelnaam RENAME nieuwabelnaam;
```

Opgave 39 Open het SQL-venster en voer de volgende query uit:

```
ALTER TABLE leerlingen RENAME studenten;
```

6.3.26 MODIFY COLUMN

Met de `MODIFY COLUMN`-opdracht kunnen we de definitie (datatype) van een bestaande kolom wijzigen. Syntaxis:

```
ALTER TABLE tabelnaam MODIFY COLUMN kolomnaam datatype;
```

Opgaven 40 Open het SQL-venster en voer de volgende query uit:

```
ALTER TABLE studenten MODIFY COLUMN postcode varchar(10);
```

41 Open het SQL-venster en voer de volgende opdracht uit:

Zet kolom postcode terug naar VARCHAR(6)

Zet kolom email naar VARCHAR(30)

Voer de volledige e-mailadressen in.

6.3.27 ADD COLUMN

Met de `ADD COLUMN`-opdracht kunnen we een nieuwe kolom aan een tabel toevoegen. Syntaxis:

```
ALTER TABLE tabelnaam ADD COLUMN niewekolomnaam datatype;
```

Opgave 42 Open het SQL-venster en voer de volgende query uit:

```
ALTER TABLE studenten ADD COLUMN testkolom varchar(10);
```

6.3.28 DROP COLUMN

Met de `DROP COLUMN`-opdracht kunnen we een bestaande kolom uit een tabel verwijderen. Syntaxis:

```
ALTER TABLE tabelnaam DROP COLUMN kolomnaam;
```

Opgave 43 Open het SQL-venster en voer de volgende query uit:

```
ALTER TABLE studenten DROP COLUMN testkolom;
```

44 Open het SQL-venster en voer de volgende opdracht uit: Voeg de nieuwe kolom `klas` aan de studententabel toe. Deel de twee studenten in klas 1AO99.”

De resultaatverzameling moet er zo uitzien:

Query results operations-										
aak Print overzicht (met volledige teksten) Exporteer										
<input type="button" value="Toon :"/> 30 rijen beginnend bij 0										
taal										
Leerlingnummer	Achternaam	Voornaam	Leeftijd	Adres	Postcode	Woonplaats	Telefoonnummer	email	Opleiding	klas
1002	Huisden	Dylan	18	Middenweg 11	1008VV	Amsterdam	0204445555	dhuisden@rocva.nl	ict	1AO99
1003	Bosman	Nitin	17	Leidseweg 22	9900BB	Amstelveen	0209994444	nbosman@hotmail.com	ict	1AO99

Figuur 6.34 De resultaatverzameling

6.3.29 CHANGE COLUMN

Met de `CHANGE COLUMN`-opdracht kunnen we de naam van een bestaande kolom wijzigen. Syntaxis:

```
ALTER TABLE tabelnaam CHANGE COLUMN oudekolomnaam
nieuwekolomnaam datatype;
```

Opgave 45 Open het SQL-venster en voer de volgende query uit:

```
ALTER TABLE studenten CHANGE COLUMN leerlingnummer
studentnummer INT(4);
```

46 Open het SQL-venster en maak de volgende query: “Wijzig de namen van alle kolommen zodat alle namen beginnen met een hoofdletter, bijvoorbeeld studentnummer moet Studentnummer zijn.”

6.3.30 ADD PRIMARY KEY

Met de `ADD PRIMARY KEY`-opdracht kunnen we een primary key voor een tabel definiëren. Syntaxis:

```
ALTER TABLE tabelnaam ADD PRIMARY KEY(kolomnaam);
```

Opgave 47 Open het SQL-venster en voer de volgende query uit:

```
ALTER TABLE studenten ADD PRIMARY KEY(studentnummer);
```

6.3.31 DROP PRIMARY KEY

Met de `DROP PRIMARY KEY`-opdracht kunnen we de primary key van een tabel verwijderen. Syntaxis:

```
ALTER TABLE tabelnaam DROP PRIMARY KEY;
```

Opgaven 48 Open het SQL-venster en voer de volgende query uit:

```
ALTER TABLE studenten DROP PRIMARY KEY;
```

49 Open het SQL-venster en maak de volgende query: "Zet de studentnummerkolom weer als primary key."

MySQL code camp - Lab 1

DBadministrator

Open de albumtabel van de mp3shop database en voeg een nieuw veld eraan toe:

Veldnaam: voorraad

Datatype: INT(3)

Voeg de voorraadwaarden toe uit onderstaande figuur. De resultaatverzameling moet er dan zo uitzien:

Figuur 6.35

De resultaatverzameling met de voorraadgegevens

	albumcode	titel	artiest	genre	prijs	voorraad
	X A1303	Cafe Atlantico	Cesarie Evora	World	3.00	100
	X A1101	Rumba Azul	Caetano Veloso	Latin	4.90	50
	X A1404	Survivor	Destiny's Child	R&B	3.00	789
	X A1606	Oh Girl	The Chi-Lites	Pop	3.00	2
	X A1505	Der Her ist mein getreue	Ton Koopman	Klasiek	5.50	30
	X A1999	Closing Time	Tom Waits	Rock	3.00	0
	X A1123	Irresistible	Celia Cruz	Latin	3.50	23
	X A1124	Marvin Gaye II	Marvin Gaye	R&B	4.00	154
	X A1321	Mi Sangre	Juanes	Latin	3.90	123
	X A2001	Greatest Hits 2	Queen	Rock	3.00	0
	X A2003	3121	Prince	Rock	3.45	0
	X A2444	Antologia I	Paco de Lucia	World	3.00	320

Below the table, there is a note: 'Selecteer alles / Deselecteer alles Met geselecteerd: '. There are also two additional 'Toon' buttons at the bottom of the page.

6.4 MySQL stringfuncties

MySQL heeft een aantal stringfuncties om de gegevens uit de databasetabellen te kunnen verwerken:

6.4.1 CONCAT()

De functie `CONCAT()` krijgt twee of meer inputparameters. Het resultaat is een string met de aaneengesloten input parameters. Als een van de inputparameters `NULL` is, dan is het resultaat ook `NULL`. Syntaxis:

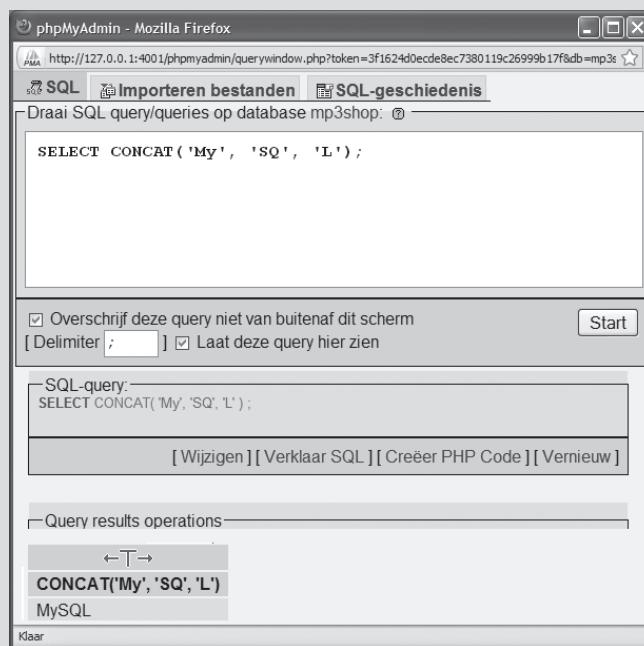
```
CONCAT(str1,str2,...);
```

- Opgave 50** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT CONCAT('My', 'SQ', 'L');
```

Het resultaat is: MySQL

Figuur 6.36
De concat-functie



6.4.2 CONCAT_WS()

De functie `CONCAT _ WS()` is een speciale vorm van `CONCAT()`. De eerste inputparameter `ws` staat voor ‘with separator’, ofwel ‘met scheidingsteken.’ Het resultaat is een string met de aaneengesloten inputparameters met een scheidingsteken. Syntaxis:

```
CONCAT _ WS(ws,str1,str2,...);
```

- Opgave 51** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT CONCAT _ WS(':', 'Naam', 'Adres', 'Woonplaats');
```

Resultaat is: Naam:Adres:Woonplaats

6.4.3 CHAR_LENGTH()

De functie `CHAR_LENGTH()` geeft de lengte van de inputstrings terug.
Syntaxis:

```
CHAR_LENGTH(str);
```

- Opgave 52** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT CHAR_LENGTH('MySQL');
```

Resultaat is: 5

6.4.4 FIND_IN_SET()

De functie `FIND_IN_SET()` geeft de positie van de gezochte string in de stringlijst. Bij de parameters is `str` is de gezochte string en `strlijst` een lijst met strings. Syntaxis:

```
FIND_IN_SET(str, strlijst);
```

- Opgave 53** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT FIND_IN_SET('Adres', 'Naam,Adres,Woonplaats');
```

Het resultaat is: 2

6.4.5 FORMAT()

De functie `FORMAT()` formateert het getal `G` naar een formaat zoals `#,##,##.##` en `D` is het aantal decimale posities. Als `D` gelijk is aan 0, dan krijgt het getal geen decimale posities. Syntaxis:

```
FORMAT(G,D);
```

- Opgave 54** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT FORMAT(1234.1234567, 3);
```

Het resultaat is: 1234.123

6.4.6 INSERT()

De functie `INSERT()` voegt in de oorspronkelijke string `str` de `nieuwestr` in, op positie `pos` en met lengte `len`.
Syntaxis:

```
INSERT(str,pos,len,nieuwstr);
```

- Opgave 55** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT INSERT('drievoud',1,4, 'vier');
```

Het resultaat is: viervoud

6.4.7 INSTR()

De functie `INSTR()` geeft de positie van de eerste gevonden `substr` in `str`. Syntaxis:

```
INSTR(str,substr);
```

- Opgave 56** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT INSTR('drievoud', 'voud');
```

Het resultaat is: 5

6.4.8 LEFT()

De functie `LEFT()` geeft het linker deel met een lengte `len` van de string `str`. Syntaxis:

```
LEFT(str,len);
```

- Opgave 57** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT LEFT('drievoud', 4);
```

Het resultaat is: 'drie'

6.4.9 LOCATE()

De functie `LOCATE()` zoekt de string `substr` in de string `str` en geeft de positie terug. Je kunt ook de beginpositie van de zoekopdracht aangeven. Syntaxis:

```
LOCATE(substr,str);
```

- Opgaven 58** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT LOCATE('drie', 'drie maal drie');
```

Het resultaat is: 1

- 59** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT LOCATE('drie', 'drie maal drie',5);
```

Het resultaat is: 11

6.4.10 LOWER()

De functie `LOWER()` zet inputstring om naar kleine letters. Syntaxis:

```
LOWER(str);
```

- Opgave 60** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT LOWER('NAAM, ADRES, WOONPLAATS');
```

Het resultaat is: naam,adres,woonplaats

6.4.11 LPAD()

De functie `LPAD()` voegt links aan de string `str` de vulling `padstr` toe.

Het resultaat heeft een lengte `len`. Syntaxis:

```
LPAD(str,len,padstr);
```

- Opgave 61** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT LPAD('1234',6,'$$');
```

Het resultaat is: ' \$\$1234'

6.4.12 LTRIM()

De functie `LTRIM()` verwijdert alle linker spaties uit `str`. Syntaxis:

```
LTRIM(str);
```

- Opgave 62** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT LTRIM('      1234');
```

Het resultaat is: '1234'

6.4.13 POSITION()

De functie `POSITION()` geeft de positie van de substring `substr` in string `str`. Syntaxis:

```
POSITION(substr IN str);
```

- Opgave 63** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT POSITION('voud' IN 'honderdvoud');
```

Het resultaat is: 8

6.4.14 REPEAT()

De functie `REPEAT()` herhaalt de inputstring een aantal keren. Syntaxis:

```
REPEAT(str, aantalkeer);
```

- Opgave 64** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT REPEAT('=',15);
```

Het resultaat is: ‘=====’

6.4.15 REPLACE()

De functie `REPLACE()` vervangt in de inputstring het gedeelte `vervang str` door het gedeelte `met str`. Syntaxis:

```
REPLACE(input str, vervang str, met str);
```

- Opgave 65** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT REPLACE('januari 3 t/m januari 7', 'januari', 'februari');
```

Het resultaat is: ‘februari 3 t/m februari 7’

6.4.16 REVERSE()

De functie `REVERSE()` keert de volgorde in de inputstring om. Syntaxis:

```
REVERSE(input str);
```

- Opgave 66** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT REVERSE('1234567890');
```

Het resultaat is: ‘0987654321’

6.4.17 RIGHT()

De functie `RIGHT()` retourneert de rechter `len` tekens van de string `str`. Syntaxis:

```
RIGHT(str,len);
```

- Opgave 67** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT RIGHT('1234567890',3);
```

Het resultaat is: ‘890’

6.4.18 RPAD()

De functie `RPAD()` voegt links aan de string `str` de vulling `padstr` toe. Het resultaat is `len` posities lang. Syntaxis:

```
RPAD(str,len,padstr);
```

- Opgave 68** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT RPAD('1234',6,'--');
```

Het resultaat is: '1234--'

6.4.19 RTRIM()

De functie `RTRIM()` verwijdert alle rechterspaties van de string `str`.

Syntaxis:

```
RTRIM(str);
```

- Opgave 69** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT RTRIM('1234      ');
```

Het resultaat is: '1234'

6.4.20 SPACE()

De functie `SPACE()` retourneert een string met `N` spaties. Syntaxis:

```
SPACE(N);
```

- Opgave 70** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT SPACE(5);
```

Het resultaat is: ' '

6.4.21 SUBSTRING(FROM)

De functie `SUBSTRING(FROM)` retourneert de substring van string `str` beginnend op positie `pos`. Syntaxis:

```
SUBSTRING(str FROM pos);
```

- Opgave 71** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT SUBSTRING('phpMyAdmin' FROM 6);
```

Het resultaat is: 'Admin'

6.4.22 SUBSTRING(FROM FOR)

De functie `SUBSTRING(FROM FOR)` retourneert de substring van string `str` beginnend op positie `pos` en met een lengte van `len` posities. Syntaxis:

```
SUBSTRING(str FROM pos FOR len);
```

- Opgave 72** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT SUBSTRING('phpMyAdmin' FROM 6 FOR 2);
```

Het resultaat is: ‘Ad’

6.4.23 SUBSTRING_INDEX()

De functie `SUBSTRING _ INDEX()` retourneert de substring van string `str` beginnend op positie 1 t/m het geval `N` van het indexteken. Syntaxis:

```
SUBSTRING _ INDEX(str, indexteken, N);
```

- Opgave 73** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT SUBSTRING _ INDEX('Naam:Adres:Woonplaats', ':',2);
```

Het resultaat is: ‘Naam:Adres’

Als `N` negatief is, dan wordt de substring vanaf de rechterkant genomen.

```
SELECT SUBSTRING _ INDEX('Naam:Adres:Woonplaats', ':',-2);
```

Resultaat is: ‘Adres:Woonplaats’

6.4.24 TRIM()

De functie `TRIM()` verwijdert alle spaties aan het begin en eind (‘leading’ en ‘trailing’ spaties) van de string `str`. Syntaxis:

```
TRIM(str);
```

- Opgave 74** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT TRIM('    MYSQ    ');
```

Het resultaat is: ‘MSQL’

6.4.25 TRIM(LEADING FROM)

De functie `TRIM(LEADING FROM)` verwijdert aan het begin van de string `str` alle exemplaren van een opgegeven teken. Syntaxis:

```
TRIM(LEADING leidende-tekens FROM str);
```

- Opgave 75** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT TRIM(LEADING '$' FROM '$$12345');
```

Het resultaat is: ‘12345’

6.4.26 TRIM(TRAILING FROM)

De functie TRIM(TRAILING FROM) verwijdert aan het eind van de string str alle exemplaren van een opgegeven teken. Syntaxis:

```
TRIM(TRAILING eindende-tekens FROM str);
```

- Opgave 76** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT TRIM(TRAILING '-' FROM '12345--');
```

Het resultaat is: '12345'

6.4.27 UPPER()

De functie UPPER() wijzigt de inhoud van de string str in hoofdletters. Syntaxis:

```
UPPER(str);
```

- Opgave 77** Open het SQL-venster in phpMyAdmin en voer de volgende MySQL-opdracht uit:

```
SELECT UPPER('MySql');
```

Het resultaat is: 'MYSQL'

MySQL code camp - Lab 2

DBadministrator

```
SET @tekst = "
```

De database administrator (DBA) is verantwoordelijk voor het ontwerpen, implementatie en onderhoud van de databases binnen een organisatie. De rol van de DBA is het monitoren en verbeteren van de database performance en veiligheid.

Voer de juiste stringfuncties in om de volgende actie uit te voeren:

1. De substring DBA moet vervangen door "dba"
2. Geef de lengte van de hele tekst.
3. Geef het aantal keer dat het woord database in de tekst voorkomt.
4. Voeg aan het einde van de tekst het woord "EINDE" toe.

6.5 Stored programma's in MySQL-databases

Stored programma's zijn databaseobjecten. Deze objecten coderen we in de SQL-querytaal. Deze programma's kunnen we in de databaseserver opslaan en we kunnen ze op een later tijdstip vanuit de server aanroepen. Stored programma's kunnen procedures, functies, views of triggers zijn.

Waarvoor gebruiken we stored programma's? Stored programma's zijn nuttig voor multiplatform-applicaties, het verminderen van netwerkverkeer en voor beveiliging van data. Voorbeelden zijn:

- Verschillende klantapplicaties (webapplicaties) die geschreven zijn door programmeurs in verschillende programmeertalen kunnen gebruik maken van stored SQL-programma's.
- Stored programma's worden volledig uitgevoerd op de dataserver, zodat er minder dataverkeer plaats vindt tussen de klant (de webserver) en de dataserver.
- Banken, bijvoorbeeld, gebruiken stored programma's voor dataverkeer-operaties. Dit gebeurt dan in een veilige en consistente omgeving waar de bank volledig controle heeft over de operaties.

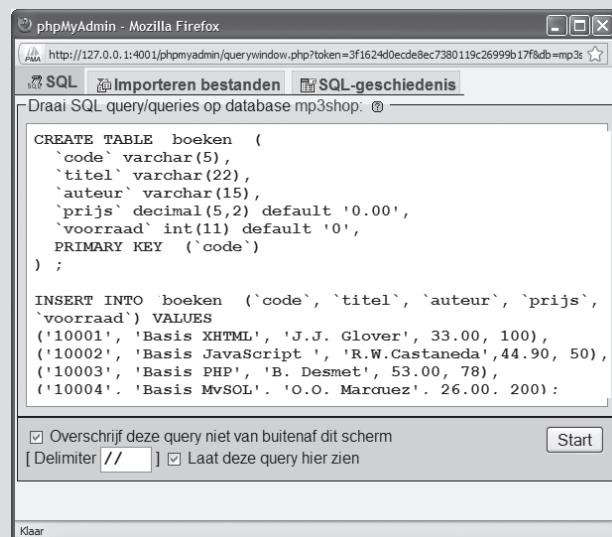
Een nadeel is dat de server kan overbelast raken. Dit kan een probleem zijn als op een gegeven moment veel webservers queries sturen naar één enkele dataserver. Dit probleem wordt meestal opgelost door meer data-servers in te schakelen.

Opgave 78 Maak nu eerst een oefentabel. Open het SQL-venster en voer de volgende MySQL-opdracht uit:

```
CREATE TABLE boeken (
  'code' varchar(5),
  'titel' varchar(22),
  'auteur' varchar(15),
  'prijs' decimal(5,2) default '0.00',
  'voorraad' int(11) default '0',
  PRIMARY KEY ('code')
);

INSERT INTO boeken ('code', 'titel', 'auteur', 'prijs',
'voorraad') VALUES
('10001', 'Basis XHTML', 'J.J. Glover', 33.00, 100),
('10002', 'Basis JavaScript ', 'R.W.Cstaneda', 44.90, 50),
('10003', 'Basis PHP', 'B. Desmet', 53.00, 78),
('10004', 'Basis MySQL', 'O.O. Marquez', 26.00, 200);
```

Figuur 6.37
SQL-venster met
create boekentabel



Het resultaat is de volgende boekentabel:

Figuur 6.38

SQL-venster met
create boekentabel

← ↑ →	code	titel	auteur	prijs	voorraad
<input type="checkbox"/>	10001	Basis XHTML	J.J. Glover	33.00	100
<input checked="" type="checkbox"/>	10002	Basis JavaScript	R.W.Castaneda	44.90	50
<input type="checkbox"/>	10003	Basis PHP	B. Desmet	53.00	78
<input type="checkbox"/>	10004	Basis MySQL	Q.Q. Marquez	26.00	200

6.5.1 SQL-variabelen declareren

SQL-variabelen gebruiken we om waarden tijdelijk te bewaren. Syntaxis:

```
SET @variabele = waarde;
```

In het volgende voorbeeld maken we de twee variabelen @melding en @error als volgt:

```
SET @melding = 'Fout opgetreden';
SET @error = 'Print error';
```

Met de SELECT-opdracht kunnen we de waarden van variabelen zien.

Bijvoorbeeld:

```
SELECT @error;
```

6.5.2 CREATE PROCEDURE

De CREATE PROCEDURE-opdracht maakt een nieuwe procedure aan.

Syntaxis:

```
CREATE PROCEDURE proc_naam()
BEGIN
Procedure definitie;
END;
//
```

Het laatste teken (//) is het ‘delimiter’-teken, dat het einde aangeeft van de query-opdracht. Je mag dit teken zelf bepalen.

6.5.3 Procedure parameters

Als je parameters in een procedure gebruikt, moet je aangeven of het IN-, OUT- of INOUT-parameters zijn.

- IN parameters verwerk je binnen de procedure, maar de oorspronkelijke waarde wordt *niet* gewijzigd.
- INOUT parameters verwerk je binnen de procedure, maar de oorspronkelijke waarde wordt *wel* gewijzigd.
- OUT parameters creëer je binnen de procedure.

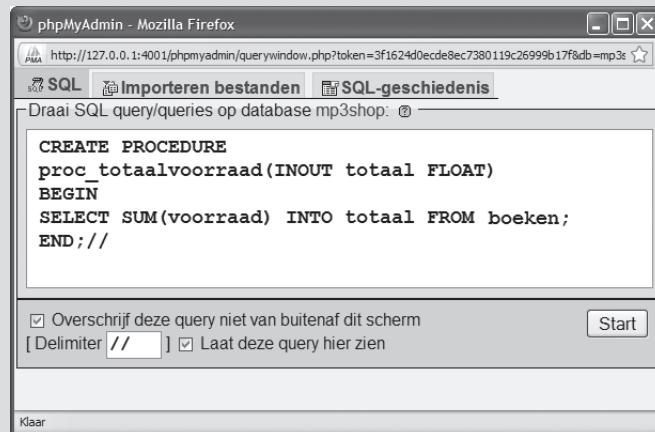
Je moet ook aangeven welk datatype de parameters hebben. Syntaxis:

```
CREATE PROCEDURE proc_naam(IN param1 INT, OUT param2
VARCHAR(5))
```

Opgaven 79 In deze **Opgave** maak je een CREATE PROCEDURE-opdracht met een INOUT parameter. Open het SQL-venster en voer de volgende MySQL-opdracht uit:

```
CREATE PROCEDURE
proc _ totaalvoorraad(INOUT totaal FLOAT)
BEGIN
SELECT SUM(voorraad) INTO totaal FROM boeken;
END; //
```

Figuur 6.39
SQL-venster met
create procedure

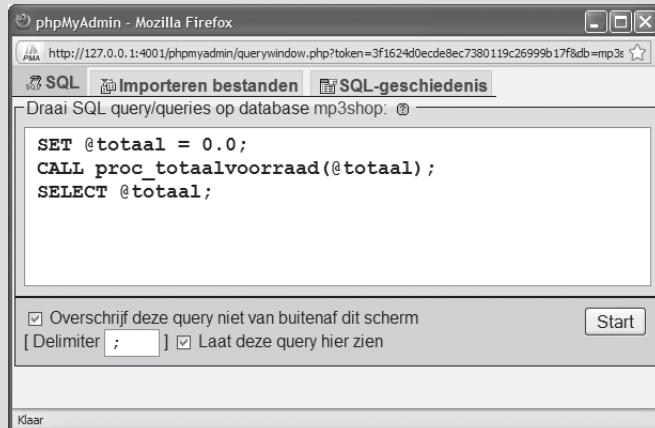


Aan het einde van je CREATE PROCEDURE-opdracht geef je in plaats van (;) een ander delimiter-teken, zoals //. Het gekozen delimiter-teken typ je onderaan, dan klik je op Start.

Om de procedure aan te roepen open je het SQL-venster en voer je de volgende MySQL-opdrachten in:

```
SET @totaal = 0.0;
CALL proc _ totaalvoorraad(@totaal);
SELECT @totaal;
```

Figuur 6.40
SQL-venster met
call procedure



Typ het delimiter-teken (;) onderaan, daarna klik je op Start.

Figuur 6.41

Resultaat van de
call procedure

← T →
@totaal
428

- Opgave 80** Open het SQL-venster en voer de volgende opdracht uit: Maak een nieuwe procedure met de naam proc_minprijs. Deze procedure selecteert de laagste prijs uit de boekentabel.

Tip: gebruik de SELECT MIN-clausule.

Roep je proc_minprijs() procedure als volgt aan:

```
SET @minprijs = 0.0;
CALL proc_minprijs(@minprijs);
SELECT FORMAT(@minprijs,2);
```

Het resultaat is:

Figuur 6.42

Resultaat van de
call procedure
maxprijs

← T →
FORMAT(@minprijs,2)
26.00

6.5.4 DROP PROCEDURE

De `DROP PROCEDURE`-opdracht verwijdert een bestaande procedure.

Syntaxis:

```
DROP PROCEDURE proc_naam;
```

- Opgave 81** Open het SQL-venster en voer een query-opdracht uit. Maak een nieuwe procedure als volgt:

```
CREATE PROCEDURE
proc_watismaxprijs(INOUT maxprijs FLOAT, OUT antwoord
VARCHAR(255))
```

Daarna roep je de procedure als volgt aan:

```
CALL proc_watismaxprijs(@maxprijs,@antwoord);
SELECT @antwoord;
```

Figuur 6.43

Resultaat van de
call procedure
watismaxprijs

← T →
@antwoord
De maximale prijs voor een boek is: 53

Tip: gebruik voor het resultaat voor het resultaat de `CONCAT` tekstfunctie.

6.5.5 SHOW

De `SHOW`-opdracht geeft de definitie van een bestaande procedure weer.

Syntaxis:

```
SHOW CREATE PROCEDURE proc_naam;
```

Opgaven 82 Open het SQL-venster en voer de volgende MySQL-opdracht uit:

```
SHOW CREATE PROCEDURE proc_watismxprijs;
```

Figuur 6.44

Resultaat van SHOW procedure proc_watismxprijs

Procedure	sql_mode	Create Procedure
proc_watismxprijs		CREATE DEFINER='root'@'localhost' PROCEDURE `proc_watismxprijs`(INOUT maxprijs FLOAT, OUT antwoord VARCHAR(255)) BEGIN SELECT MAX(prijs) INTO maxprijs FROM boeken; SET antwoord = concat("De maximale prijs voor een boek is: ", maxprijs); END

83 Open het SQL-venster en maak een SHOW-opdracht voor de procedure proc_totaalvoorraad.**6.5.6 IF**

Met de IF-opdracht kunnen we beslissingstructuren binnen een procedure coderen. Syntaxis:

```
IF voorwaarde
    THEN
        acties uit te voeren;
END IF;
```

Opgaven 84 Open het SQL-venster en voer de volgende query-opdracht uit:

```
CREATE PROCEDURE proc_checkparams(
    INOUT param1 VARCHAR(5),
    INOUT param2 INT,
    OUT melding VARCHAR(50))
SQL SECURITY DEFINER
BEGIN
    IF param1 = '' OR param2 = '' THEN
        SET melding = 'Foutmelding: er zijn een of meer input errors';
    ELSE
        SET melding = 'Er zijn geen input errors';
    END IF;
END //
```

Open het SQL-venster en voer de volgende query-opdracht uit:

```
SET @param1 = '';
SET @param2 = 0;
SET @melding = '';
CALL proc_checkparams(@param1,@param2,@melding);
```

Figuur 6.45

Resultaat van procedure proc_checkparams

←T→
@melding
Foutmelding: er zijn een of meer input errors

85 Open het SQL-venster en verander de proc_checkparams procedure naar drie INOUT-parameters.

6.5.7 CASE

De CASE beslissingstructuur codeer je als volgt:

```
CASE variabele
    WHEN waarde1 THEN acties uit te voeren . . . ;
    WHEN waarde2 THEN acties uit te voeren . . . ;
    ELSE acties uit te voeren . . . ;
END CASE;
```

Opgaven 86 Open het SQL-venster en voer de volgende query-opdracht uit:

```
CREATE PROCEDURE proc_setvoorraad(IN code INT)
BEGIN
    CASE code
        WHEN 0 THEN UPDATE boeken SET voorraad = 0;
        WHEN 1 THEN UPDATE boeken SET voorraad = 100;
        WHEN 2 THEN UPDATE boeken SET voorraad = 200;
        ELSE UPDATE boeken SET voorraad = 300;
    END CASE;
END;
//
```

Open het SQL-venster en voer de volgende query-opdracht uit:

```
SET @code = 0;
call proc_setvoorraad(@code);
```

Figuur 6.46

Resultaat van procedure proc_setvoorraad

← T →	code	titel	auteur	prijs	voorraad
<input type="checkbox"/>	10001	Basis XHTML	J.J. Glover	33.00	0
<input type="checkbox"/>	10002	Basis JavaScript	R.W.Castaneda	44.90	0
<input type="checkbox"/>	10003	Basis PHP	B. Desmet	53.00	0
<input type="checkbox"/>	10004	Basis MySQL	Q.Q. Marquez	26.00	0

87 Open het SQL-venster, voer de proc_setvoorraad procedure uit en stel de voorraad in op 300.

6.5.8 WHILE...END WHILE

Met de WHILE-opdracht kunnen we herhaalstructuren coderen. Syntaxis:

```
WHILE voorwaarde DO
    Acties uit te voeren;
END WHILE;
```

Opgaven 88 Open het SQL-venster en voer de volgende query-opdracht uit:

```
CREATE PROCEDURE proc_voegdrie codes()
BEGIN
    DECLARE controlvar INT DEFAULT 0;
    DECLARE boekcode VARCHAR(5);
    SELECT MAX(code) INTO boekcode FROM boeken;
    WHILE controlvar < 3 DO
        SET boekcode = boekcode + 1;
        INSERT INTO boeken VALUES(boekcode, " ", " ", 0,0);
        SET controlvar = controlvar + 1;
    END WHILE;
END//
```

Figuur 6.47

Resultaat van procedure proc_voegdriecodes

← T →	code	titel	auteur	prijs	voorraad
<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	10001	Basis XHTML	J.J. Glover	33.00	300
<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	10002	Basis JavaScript	R.W.Castaneda	44.90	300
<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	10003	Basis PHP	B. Desmet	53.00	300
<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	10004	Basis MySQL	Q.Q. Marquez	26.00	300
<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	10006			0.00	0
<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	10007			0.00	0
<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	10005			0.00	0

- 89** Open het SQL-venster en maak alsvolgt de procedure proc_boektoevoegen:

```
CREATE PROCEDURE proc_boektoevoegen(@titel,@auteur,@prijs,@voorraad)
```

Met deze procedure moet je als volgt een nieuw boek kunnen toevoegen aan je boekentabel:

```
SET @titel = 'Basis CSS';
SET @auteur = 'R.W. Castaneda';
SET @prijs = 39.99;
SET @voorraad = 200;
CALL proc_boektoevoegen (@titel,@auteur,@prijs,@voorraad);
```

Tip: de boekcode voor een nieuw boek is de hoogste code plus 1

Figuur 6.48

Resultaat van procedure proc_boektoevoegen

← T →	code	titel	auteur	prijs	voorraad
<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	10001	Basis XHTML	J.J. Glover	33.00	300
<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	10002	Basis JavaScript	R.W.Castaneda	44.90	300
<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	10003	Basis PHP	B. Desmet	53.00	300
<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	10004	Basis MySQL	Q.Q. Marquez	26.00	300
<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	10006			0.00	0
<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	10008	Basis CSS	R.W. Castaneda	39.99	200
<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	10007			0.00	0
<input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/>	10005			0.00	0

6.5.9 DECLARE CURSOR

SQL-cursors zijn pointers naar een specifieke rij in een tabel. Zo kunnen we met behulp van de WHILE-lus door alle rijen uit een tabel doorlopen. Cursors moet je eerst declareren met `DECLARE CURSOR`, dan openen en dan pas lezen met de `FETCH`-opdracht. Aan het einde moet je cursors sluiten. Syntaxis:

```
DECLARE cursor_naam CURSOR FOR selectopdracht;
OPEN cursor_naam;
FETCH cursor_naam INTO variabele [, variabele];
CLOSE cursor_naam;
```

6.5.10 DECLARE HANDLER

Na het doorlopen door alle rijen uit een tabel, ben je aan het einde van de tabel. Deze conditie moet behandeld worden door ‘handlers’. Een handler zegt wat moet gedaan worden in een bepaalde situatie. Syntaxis:

```
DECLARE handler_naam HANDLER FOR conditie actie;
```

In de volgende opgave treedt de conditie NOT FOUND op aan het einde van een tabel. Deze conditie gebruikt de handler CONTINUE. Deze handler voert de actie uit SET finish = 1.

```
DECLARE CONTINUE HANDLER FOR NOT FOUND SET finish = 1;
```

Opgaven 90 Open het SQL-venster en voer de volgende MySQL-opdracht uit:

```
CREATE PROCEDURE proc_verwijderblancorijen()
BEGIN
    DECLARE dezecode VARCHAR(5);
    DECLARE dezetitel VARCHAR(22);
    DECLARE finish INT DEFAULT 0;
    DECLARE cursor1 CURSOR FOR SELECT code, titel FROM boeken;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET finish = 1;
    OPEN cursor1;
    WHILE NOT finish DO
        FETCH cursor1 INTO dezecode, dezetitel;
        IF dezetitel = '' THEN
            DELETE FROM boeken WHERE code = dezecode;
        END IF;
    END WHILE;
    CLOSE cursor1;
END//
```

Figuur 6.49

Resultaat van procedure proc_verwijderblancorijen

← ↑ →	code	titel	auteur	prijs	voorraad
<input type="checkbox"/> <input type="pen"/> <input checked="" type="cross"/>	10001	Basis XHTML	J.J. Glover	33.00	300
<input type="checkbox"/> <input type="pen"/> <input checked="" type="cross"/>	10002	Basis JavaScript	R.W.Castaneda	44.90	300
<input type="checkbox"/> <input type="pen"/> <input checked="" type="cross"/>	10003	Basis PHP	B. Desmet	53.00	300
<input type="checkbox"/> <input type="pen"/> <input checked="" type="cross"/>	10004	Basis MySQL	Q.Q. Marquez	26.00	300
<input type="checkbox"/> <input type="pen"/> <input checked="" type="cross"/>	10008	Basis CSS	R.W. Castaneda	39.99	200

91 Open het SQL-venster en maak als volgt een nieuwe procedure:

```
CREATE PROCEDURE proc_nieuwevoorraad(IN nieuwevoorraad INT)
```

Deze procedure leest alle rijen in de boekentabel. Als de voorraad onder de 300 is, maakt de procedure een UPDATE-opdracht: de voorraad wordt geüpdatet naar nieuwevoorraad.

Roep de procedure als volgt aan:

```
SET @nieuevevoorraad = 500;
CALL proc_nieuwevoorraad(@nieuevevoorraad);
```

Figuur 6.50

Resultaat van functie func_prijsverlaging

← ↑ →	code	titel	auteur	prijs	voorraad
<input type="checkbox"/> <input type="pen"/> <input checked="" type="cross"/>	10001	Basis XHTML	J.J. Glover	33.00	300
<input type="checkbox"/> <input type="pen"/> <input checked="" type="cross"/>	10002	Basis JavaScript	R.W.Castaneda	44.90	300
<input type="checkbox"/> <input type="pen"/> <input checked="" type="cross"/>	10003	Basis PHP	B. Desmet	53.00	300
<input type="checkbox"/> <input type="pen"/> <input checked="" type="cross"/>	10004	Basis MySQL	Q.Q. Marquez	26.00	300
<input type="checkbox"/> <input type="pen"/> <input checked="" type="cross"/>	10008	Basis CSS	R.W. Castaneda	39.99	500

6.5.11 CREATE FUNCTION

Een functie is bijna hetzelfde als een procedure. Het enige verschil is dat functies de RETURN-clausule gebruiken. Bij functies hoef je ook niet aan te geven of een input parameter IN, OUT of INOUT is. Syntaxis:

```
CREATE FUNCTION func _ naam()
RETURN datatype
DETERMINISTIC
BEGIN
    functiedefinitie;
    RETURN(waarde);
END//
```

Opgaven 92 Open het SQL-venster en voer de volgende MySQL-opdracht uit:

```
CREATE FUNCTION func _ prijsverlaging(boekcode INT, procent
FLOAT)
RETURNS INT
DETERMINISTIC
BEGIN
DECLARE dezecode FLOAT;
DECLARE dezeprijs FLOAT;
DECLARE verlaging FLOAT;
DECLARE finish INT DEFAULT 0;
DECLARE cursor1 CURSOR FOR SELECT code, prijs FROM boeken;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET finish = 1;
OPEN cursor1;
WHILE NOT finish DO
    FETCH cursor1 INTO dezecode, dezeprijs;
    IF dezecode = boekcode THEN
        SET verlaging = dezeprijs * (procent / 100);
        UPDATE boeken SET prijs = dezeprijs - verlaging WHERE
code = boekcode;
    END IF;
END WHILE;
CLOSE cursor1;
RETURN(1);
END//
```

Om de prijs van het boek met code 10004 met 10 procent te verlagen, roep je de functie als volgt aan:

```
SELECT func _ prijsverlaging('10004', 10);
```

Figuur 6.51

Resultaat van functie func _ prijsverlaging

← ↑ →	code	titel	auteur	prijs	voorraad
<input type="checkbox"/> <input type="pen"/> <input type="cross"/>	10001	Basis XHTML	J.J. Glover	33.00	300
<input type="checkbox"/> <input type="pen"/> <input type="cross"/>	10002	Basis JavaScript	R.W.Castaneda	44.90	300
<input type="checkbox"/> <input type="pen"/> <input type="cross"/>	10003	Basis PHP	B. Desmet	53.00	300
<input type="checkbox"/> <input type="pen"/> <input type="cross"/>	10004	Basis MySQL	Q.Q. Marquez	23.40	300

93 Open het SQL-venster en wijzig de functie func _ prijsverlaging als volgt:

```
CREATE FUNCTION func _ prijsverlaging(overdeprijs FLOAT,
procent FLOAT)
```

De functie voert een prijsverlaging met het opgegeven percentage alleen uit als de prijs boven de opgegeven prijs `overdeprijs` ligt.

Als je een prijsverlaging van 10 procent wilt doorvoeren voor boeken boven de 40.00 euro, roep je de functie als volgt aan:

```
SELECT func_prijsverlaging(40, 10);
```

Het resultaat moet er als volgt uitzien:

Figuur 6.52

Resultaat van functie `func_prijsverlaging`

←→	code	titel	auteur	prijs	voorraad
<input type="checkbox"/>	10001	Basis XHTML	J.J. Glover	33.00	300
<input type="checkbox"/>	10002	Basis JavaScript	R.W. Castaneda	40.41	300
<input type="checkbox"/>	10003	Basis PHP	B. Desmet	47.70	300
<input type="checkbox"/>	10004	Basis MySQL	Q.Q. Marquez	23.40	300
<input type="checkbox"/>	10008	Basis CSS	R.W. Castaneda	39.99	500

6.5.12 CREATE VIEW

Een view is een soort *virtuele tabel*, want een view is gebaseerd op queries op andere tabellen. Een view is een ‘dynamische opname’ van een tabel. Wanneer de gegevens in de tabel zijn gewijzigd, veranderen ook de gegevens in de view. Syntaxis:

```
CREATE VIEW view_naam kolom_naam [, kolom_naam]
AS SELECT-opdracht
```

Met `CREATE OR REPLACE` maak je een nieuwe view of vervang je een oude view als die al bestaat.

Met de `SELECT`-opdracht kun je selecteren uit tabellen of views.

Opgaven 94 Open het SQL-venster en voer de volgende MySQL-opdracht uit:

```
CREATE VIEW boekenview AS SELECT * FROM boeken;
```

Deze view roep je als volgt aan:

```
SELECT * FROM boekenview;
```

95 Open het SQL-venster en maak de `view_voordeel` voor alle boeken waarvoor de prijs onder de 30.00 euro ligt.

6.5.13 CREATE TRIGGER

Een trigger is een databaseobject dat geassocieerd is met een tabel. Triggers worden geactiveerd door events. Een event is een geprogrammeerde gebeurtenis in je tabel. Bijvoorbeeld tabel UPDATES of INSERTS zijn events. Syntaxis:

```
CREATE TRIGGER trigger_naam
BEFORE [AFTER] event
ON table_naam
FOR EACH ROW
BEGIN
    definitie;
END;
```

Opgave 96 Open het SQL-venster en voer de volgende MySQL-opdracht uit:

```
CREATE TRIGGER trig_voorraadwaarschuwing
BEFORE UPDATE
ON boeken
FOR EACH ROW
BEGIN
    IF NEW.voorraad <=10 THEN
        SET NEW.voorraad = 100;
    ELSEIF NEW.voorraad > 500 THEN
        SET NEW.voorraad = 1000;
    END IF;
END//
```

MySQL code comp - Lab 3

Stored programmas

Maak eerst als nieuwe functie die het aantal titels geeft met een prijs onder de opgegeven limiet:

```
CREATE FUNCTION func_prijsonder(limiet FLOAT)
```

Maak de daarna volgende view:

View-naam: view_klanten_amsterdam

View-definitie: alle klanten met de woonplaats Amsterdam

6.6 Programmering met PHP en MySQL

Je kunt pas met MySQL-databases werken nadat je een verbinding hebt gemaakt met de My_SQL-dataserver.

Stap 1: Maak een verbinding

Om een database te benaderen gebruiken we de `mysql_connect()`-functie, bijvoorbeeld:

```
$verbinding = mysql_connect("localhost", "gebruikersnaam",
                            "wachtwoord") or
die ("Kon geen verbinding maken met database");
```

Stap 2: Selecteer de database

Met de `mysql_select_db()`-functie selecteer je de database waarop je queries gaat uitvoeren, bijvoorbeeld:

```
$db = mysql_select_db("database_naam") or
die ("Kon geen database selecteren");
```

Stap 3: Definieer je query

Bijvoorbeeld:

```
$query = "SELECT * FROM tabel_naam";
```

Stap 4: Voer de query uit

De `mysql_query()`-functie geeft een pointer terug die verwijst naar de resultaatverzameling van een query:

```
$resultaat = mysql_query($query, $verbinding);
```

Let op dat er veel oorzaken zijn waardoor een query kan mislukken. Gebruik de `mysql_error()`-functie om te achterhalen waarom een bepaalde query is mislukt.

```
$resultaat = mysql_query($query, $verbinding) or die  
( mysql_error() );
```

Stap 5: Haal de query gegevens op uit de resultaatverzameling

Nadat het resultaat van een query is aangemaakt, kun je vaak de afzonderlijke rijen van de query ophalen met de functie `mysql_fetch_array()`.

```
$rij = mysql_fetch_array($resultaat, MYSQL_ASSOC)
```

Deze functie geeft de waarde `false` als er geen rijen meer opgehaald kunnen worden.

6.6.1 De SELECT FROM-opdracht

De SQL SELECT-opdracht is een zoekopdracht naar gegevens uit een tabel.

Syntaxis:

```
SELECT *  
FROM naam_tabel
```

Het `*`-teken staat voor het selecteren van alle kolommen uit de tabel.

Bijvoorbeeld: Voer de volgende query-opdracht uit:

```
SELECT * FROM album
```

Voorbeeld 6.1

Zorg ervoor dat **voorbeeld.php** er als volgt uitziet:

```
<?php  
$verbinding = mysql_connect("localhost", "root", "")  
or die ("Kon geen verbinding maken");  
mysql_select_db("mp3shop")  
or die("Kon geen database selecteren");  
$query = ("select * from album");  
$resultaat= mysql_query($query)  
or die ( mysql_error() );  
// Print tabel  
echo "<table>";  
echo "<tr bgcolor=CCCCCC>";  
if($row = mysql_fetch_array($resultaat, MYSQL_ASSOC))  
{  
    // Print headers  
    foreach($row as $key => $value )  
    {  
        echo "<th>" . $key . "</th>";  
    }  
    echo "</tr>";  
}  
$resultaat= mysql_query($query)  
or die ( mysql_error() );
```

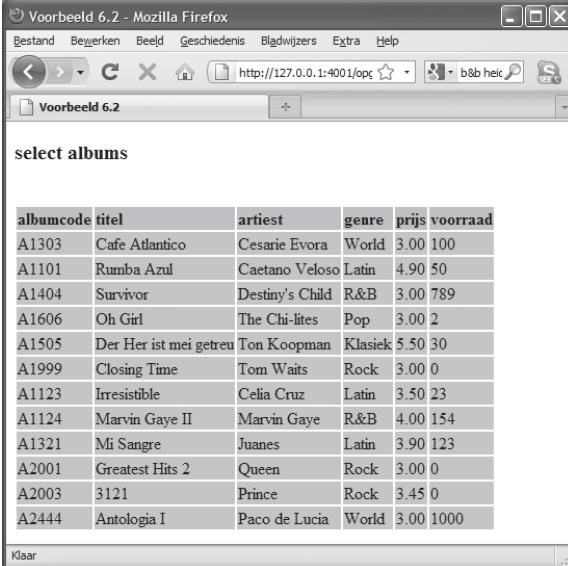
```

while ($row = mysql_fetch_array($resultaat, MYSQL_ASSOC))
{
    // Print rij
    echo "<tr bgcolor=lightblue>";
    // Zolang er nog velden zijn in $row
    while( list($key, $value) = each($row))
    {
        // Print value
        echo "<td>" . $value . "</td>";
    }
    echo "</tr>";
}
echo "</table>";
?>

```

Figuur 6.53

Het resultaat van voorbeeld 5.2 moet er als volgt uitzien



The screenshot shows a Mozilla Firefox browser window with the title 'Voorbeeld 6.2 - Mozilla Firefox'. The address bar displays 'http://127.0.0.1:4001/opg'. The main content area shows a table titled 'select albums'. The table has columns: albumcode, titel, artiest, genre, prijs, and voorraad. The data is as follows:

albumcode	titel	artiest	genre	prijs	voorraad
A1303	Cafe Atlantico	Cesarie Evora	World	3.00	100
A1101	Rumba Azul	Caetano Veloso	Latin	4.90	50
A1404	Survivor	Destiny's Child	R&B	3.00	789
A1606	Oh Girl	The Chi-lites	Pop	3.00	2
A1505	Der Her ist mei getreu	Ton Koopman	Klasiek	5.50	30
A1999	Closing Time	Tom Waits	Rock	3.00	0
A1123	Irresistible	Celia Cruz	Latin	3.50	23
A1124	Marvin Gaye II	Marvin Gaye	R&B	4.00	154
A1321	Mi Sangre	Juanes	Latin	3.90	123
A2001	Greatest Hits 2	Queen	Rock	3.00	0
A2003	3121	Prince	Rock	3.45	0
A2444	Antologia I	Paco de Lucia	World	3.00	1000

Opgave 97 Voer een query-opdracht uit die de tabel **bestelitem** weergeeft.

6.6.2 SELECT-kolommen

De SELECT-opdracht is een zoekopdracht naar specifieke kolommen in een tabel.

Syntaxis:

```

SELECT naam_kolom [,naam_kolom ]
FROM naam_tabel

```

Voer de volgende query-opdracht uit:

```

$query="SELECT artiest,titel,genre
FROM album";

```

Figuur 6.54
De resultaatverzameling moet er zo uitzien

The screenshot shows a Microsoft Internet Explorer window with the title bar 'Mijn php-script - Microsoft Internet Explorer'. The address bar contains 'http://127.0.0.1/unit4/voorbeeld.php'. The page content is titled 'select albums' and displays a table with three columns: 'artiest', 'titel', and 'genre'. The data in the table is:

artiest	titel	genre
Cesarie Evora	Cafe Atlantico	World
Caetano Veloso	Rumba Azul	Latin
Destiny's Child	Survivor	R&B
The Chi-lites	Oh Girl	Pop
Ton Koopman	Der Her ist mei getreu	Klasiek
Tom Waits	Closing Time	Rock
Celia Cruz	Irresistible	Latin
Marvin Gaye	Marvin Gaye II	R&B
Juanes	Mi Sangre	Latin
Queen	Greatest Hits 2	Rock
Prince	3121	Rock
Paco de Lucia	Antologia I	World

Opgave 98 Voer de volgende query-opdracht uit: selecteer de kolommen naam en adres vanuit de tabel Klant.

6.6.3 ORDER BY

De ORDER BY-clausule geeft een gesorteerde resultaatverzameling weer.

Voer de volgende query-opdracht uit:

```
$query="SELECT artiest,titel,genre
FROM album ORDER BY artiest";
```

Figuur 6.55
De resultaatverzameling moet er zo uitzien

The screenshot shows a Mozilla Firefox window with the title bar 'Voorbeeld 6.6.3 - Mozilla Firefox'. The address bar contains 'http://127.0.0.1:4001/opg'. The page content is titled 'Voorbeeld 6.6.3' and displays a table with three columns: 'artiest', 'titel', and 'genre'. The data in the table is identical to Figure 6.54. At the bottom of the browser window, there is a status bar with the word 'Klaar'.

artiest	titel	genre
Caetano Veloso	Rumba Azul	Latin
Celia Cruz	Irresistible	Latin
Cesarie Evora	Cafe Atlantico	World
Destiny's Child	Survivor	R&B
Juanes	Mi Sangre	Latin
Marvin Gaye	Marvin Gaye II	R&B
Paco de Lucia	Antologia I	World
Prince	3121	Rock
Queen	Greatest Hits 2	Rock
The Chi-lites	Oh Girl	Pop
Tom Waits	Closing Time	Rock
Ton Koopman	Der Her ist mei getreu	Klasiek

- Opgave 99** Voer de volgende query-opdracht uit: selecteer alle kolommen vanuit de tabel Album in titelvolgorde.

6.6.4 De WHERE-clausule

Bij de WHERE-clausule geef je een selectiecriterium aan.

Syntaxis:

```
SELECT kolomnamen FROM tabel
WHERE voorwaarde
```

Bij de WHERE-clausule kunnen we de volgende operatoren met de voorwaarde gebruiken:

Operator	Beschrijving
=	Gelijk
<>	Niet gelijk
>	Groter dan
<	Kleiner dan
>=	Groter dan of gelijk
<=	Kleiner dan of gelijk
BETWEEN	Tussen een range
LIKE	Zoek een patroon

Voer de volgende query-opdracht uit:

```
$query = "SELECT * FROM klant
WHERE woonplaats='Amsterdam' ";
```

Figuur 6.56

De resultaatverzameling moet er zo uitzien

klantnr	naam	adres	postcode	woonplaats	email
K1101	Dylan Huisden	Middenweg 11	1088VV	Amsterdam	dhuisden@roc.nl
K1102	Nitin Bosman	Leidseweg 22	9900BB	Amsterdam	nbosman@roc.nl
K1902	Tamara Kabli	Mozartstraat 22	3388JX	Amsterdam	tamka@hotmail.com

- Opgave 100** Voer de volgende query-opdracht uit: selecteer alle albums vanuit de tabel Album waarvan het genre 'Latin' is.

6.6.5 BETWEEN ... AND

De BETWEEN ... AND-operator selecteert een range tussen twee waarden. De waarden kunnen nummers, tekst of datums zijn.

Voer de volgende query-opdracht uit:

```
$query = "SELECT albumcode, titel, artiest, genre, prijs
FROM album WHERE prijs BETWEEN 3.00 AND 4.00";
```

Figuur 6.57

De resultaatverzameling moet er zo uitzien

albumcode	titel	artiest	genre	prijs
A1303	Cafe Atlantico	Cesarie Evora	World	3.00
A1404	Survivor	Destiny's Child	R&B	3.00
A1606	Oh Girl	The Chi-lites	Pop	3.00
A1999	Closing Time	Tom Waits	Rock	3.00
A1123	Irresistible	Celia Cruz	Latin	3.50
A1124	Marvin Gaye II	Marvin Gaye	R&B	4.00
A1321	Mi Sangre	Juanes	Latin	3.90
A2001	Greatest Hits 2	Queen	Rock	3.00
A2003	3121	Prince	Rock	3.45
A2444	Antologia I	Paco de Lucia	World	3.00

Opgave 101 Voer de volgende query-opdracht uit: selecteer alle bestellingen vanuit de tabel Bestelling tussen '2007-01-01' en '2007-02-01'.

6.6.6 LIMIT

De LIMIT-clausule geeft het aantal rijen in je resultaatverzameling aan.

Voer de volgende query-opdracht uit:

```
$query = "SELECT albumcode, titel, artiest, genre, prijs
FROM album WHERE prijs BETWEEN 3.00 AND 4.00 LIMIT 5";
```

Figuur 6.58

De resultaatverzameling moet er zo uitzien

albumcode	titel	artiest	genre	prijs
A1303	Cafe Atlantico	Cesarie Evora	World	3.00
A1404	Survivor	Destiny's Child	R&B	3.00
A1606	Oh Girl	The Chi-lites	Pop	3.00
A1999	Closing Time	Tom Waits	Rock	3.00
A1123	Irresistible	Celia Cruz	Latin	3.50

6.6.7 SELECT DISTINCT

De DISTINCT-clausule geeft een resultaatverzameling zonder dubbele gegevens. Bijvoorbeeld in de tabel Bestelitem heb je meerdere items met hetzelfde bestellingnr.

Voer de volgende query-opdracht uit:

```
$query = "SELECT DISTINCT bestellingnr FROM bestelitem;"
```

Figuur 6.59

De resultaatverzameling moet er zo uitzien

bestellingnr
B1201
B1203
B1204
B1205
B1206

6.6.8 INNER JOIN

We hebben een datamodel ontworpen waarin iedere tabel een primaire of verwijzende sleutel heeft. Zo kunnen we een relatie tussen de tabellen maken. Soms moeten we gegevens vanuit twee of meer tabellen selecteren. In deze gevallen gebruiken we een INNER JOIN-opdracht.

Syntaxis:

```
SELECT kolom1, kolom2, kolom3
FROM eerste_tabel
INNER JOIN tweede_tabel
ON eerste_tabel.primaire_sleutel = tweede_tabel.
verwijzende_sleutel
```

Bijvoorbeeld: Wie heeft welke bestellingen geplaatst?

Voer de volgende query-opdracht uit:

```
SELECT klant.naam, klant.email, bestelling.bestellingnr,
bestelling.datum
FROM klant
INNER JOIN bestelling
ON klant.klantnr = bestelling.klantnr
```

Figuur 6.60
De resultaatverzameling moet er zo uitzien

naam	email	bestellingnr	datum
Dylan Husden	dhusden@roc.nl	B1201	2007-01-01
Nitin Bosman	nbozman@roc.nl	B1203	2007-01-01
Joseph Demirel	josdem@hotmail.com	B1204	2007-02-15
Franco Tasyan	frantas@wanadoo.nl	B1205	2007-02-20
Tamara Kabli	tamka@hotmail.com	B1206	2007-03-13

Opgave 102 Voer de volgende SELECT JOIN-opdracht uit: Hoeveel ‘Rumba Azul’ albums zijn besteld?

Figuur 6.61
De resultaatverzameling moet er zo uitzien

titel	artiest	bestellingnr	aantal
Rumba Azul	Caetano Veloso	B1201	1
Rumba Azul	Caetano Veloso	B1203	1
Rumba Azul	Caetano Veloso	B1205	1
Rumba Azul	Caetano Veloso	B1206	1

6.6.9 LEFT JOIN

De LEFT JOIN-opdracht geeft alle rijen weer vanuit de eerste tabel, zelfs als er geen bestellingen zijn geplaatst.

Syntaxis:

```
SELECT kolom1, kolom2, kolom3
FROM eerste_tabel
LEFT JOIN tweede_tabel
ON eerste_tabel.primaire_sleutel = tweede_tabel.
verwijzende_sleutel
```

Bijvoorbeeld: Wie heeft welke bestellingen geplaatst?

```
SELECT klant.naam, klant.email, bestelling.bestellingnr,
bestelling.datum
FROM klant
LEFT JOIN bestelling
ON klant.klantnr = bestelling.klantnr
```

Figuur 6.62

De resultaatverzameling moet er zo uitzien

The screenshot shows a Microsoft Internet Explorer window with the title bar 'Mijn php-script - Microsoft Internet Explorer'. The address bar contains the URL 'http://jessika/unit4/voorbijd.php'. Below the address bar is a toolbar with various icons. The main content area displays a table with the heading 'left join select'. The table has four columns: 'naam', 'email', 'bestellingnr', and 'datum'. The data is as follows:

naam	email	bestellingnr	datum
Dylan Huisden	dhuissen@roc.nl	B1201	2007-01-01
Nitin Bosman	nbosman@roc.nl	B1203	2007-01-01
Joseph Demirel	josdem@hotmail.com	B1204	2007-02-15
Franco Tasiyan	frantasy@wanadoo.nl	B1205	2007-02-20
Akash Kabli	aka@hetnet.nl		
Tamara Kabli	tamka@hotmail.com	B1206	2007-03-13
Arnold Shaw	asha@roc.nl		

6.6.10 JOIN meerdere tabellen

Stel je wilt je database raadplegen met de volgende vraag: Welke klanten hebben het album 'Rumba Azul' besteld? Dan moet je de tabellen Klant, Bestelling, Bestelitem en Album raadplegen met een **JOIN SELECT**-opdracht.

Voer de volgende query-opdracht uit:

```
$query = "SELECT klant.naam, klant.email, album.titel,
album.artiest, bestelitem.bestellingnr, bestelitem.aantal
FROM klant
INNER JOIN (bestelling
INNER JOIN (bestelitem
INNER JOIN album
ON album.albumcode = bestelitem.albumcode)
ON bestelling.bestellingnr = bestelitem.bestellingnr)
ON klant.klantnr = bestelling.klantnr
WHERE bestelitem.albumcode = 'A1101'";
```

Figuur 6.63

De resultaatverzameling moet er zo uitzien

The screenshot shows a Microsoft Internet Explorer window with the title bar 'Mijn php-script - Microsoft Internet Explorer'. The address bar contains the URL 'http://jessika/unit4/voorbijd.php'. Below the address bar is a toolbar with various icons. The main content area displays a table with the heading 'join select vier tabellen'. The table has six columns: 'naam', 'email', 'titel', 'artiest', 'bestellingnr', and 'aantal'. The data is as follows:

naam	email	titel	artiest	bestellingnr	aantal
Dylan Huisden	dhuissen@roc.nl	Rumba Azul	Caetano Veloso	B1201	1
Nitin Bosman	nbosman@roc.nl	Rumba Azul	Caetano Veloso	B1203	1
Franco Tasiyan	frantasy@wanadoo.nl	Rumba Azul	Caetano Veloso	B1205	1
Tamara Kabli	tamka@hotmail.com	Rumba Azul	Caetano Veloso	B1206	1

6.6.11 De UPDATE-opdracht

We gebruiken de **UPDATE**-opdracht om gegevens in een tabel te wijzigen.

Syntaxis:

```
UPDATE tabel_naam
SET kolom_naam = nieuwe_waarde
WHERE voorwaarde
```

Voer de volgende query-opdracht uit:

```
$query = "UPDATE klant
SET
adres = 'Galileiplantsoen 333',
postcode = '1010RR'
WHERE klantnr = 'K1000' ";
```

Figuur 6.64

De resultaatverzameling moet er zo uitzien

The screenshot shows a Microsoft Internet Explorer window titled 'Mijn php-script - Microsoft Internet Explorer'. The address bar contains 'http://jessika/unit4/displayklant.php'. The page displays a table titled 'select klanten' with the following data:

klantnr	naam	adres	postcode	woonplaats	email
K1101	Dylan Huisden	Middenweg 11	1088VV	Amsterdam	dhuisden@roc.nl
K1102	Nitin Bosman	Leidseweg 22	9900BB	Amsterdam	nbosman@roc.nl
K1103	Joseph Demirel	Leidseplein 33	9988BB	Utrecht	josdem@hotmail.com
K1000	Franco Taslyan	Galileiplantsoen 333	1010RR	Utrecht	frantas@wanadoo.nl
K1901	Akash Kabli	Biothof 55	2299NN	Amstelveen	aka@hetnet.nl
K1902	Tamara Kabli	Mozartstraat 22	3388XX	Amsterdam	tamka@hotmail.com
K1100	Arnold Shaw	Kruislaan 1	9876FF	Rotterdam	asha@roc.nl

6.6.12 De INSERT INTO-opdracht

Je weet nu hoe je een tabel aanmaakt, maar natuurlijk wil je ook gegevens aan tabellen kunnen toevoegen. Met de `INSERT INTO`-opdracht kun je nieuwe rijen (records) aan een tabel toevoegen.

Syntaxis:

```
INSERT INTO naam_tabel
VALUES (waarde1, waarde2,...)
```

Je kunt ook kolomnamen gebruiken:

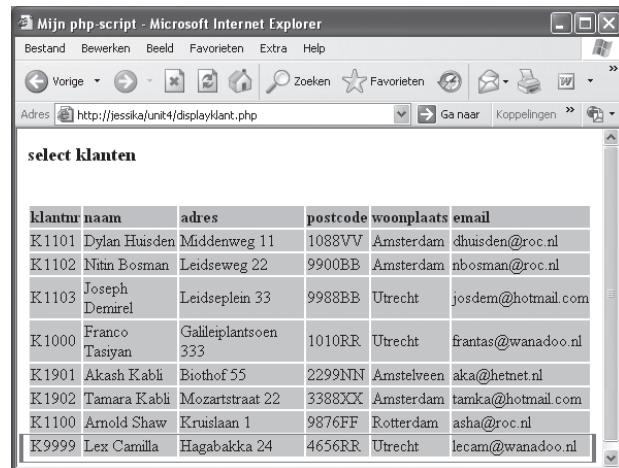
```
INSERT INTO naam_tabel
(kolom1, kolom2, kolom3...)
VALUES (waarde1, waarde2, waarde3...)
```

Voer de volgende `INSERT INTO`-opdracht uit:

```
INSERT INTO klant
(klantnr,naam,adres,postcode,woonplaats,email)
VALUES ('K9999','Lex Camilla', 'Hagabakka 24',
'4656RR','Utrecht', 'lecam@wanadoo.nl')
```

Figuur 6.65

De resultaatverzameling moet er zo uitzien



klantnr	naam	adres	postcode	woonplaats	email
K1101	Dylan Huisden	Middenweg 11	1088VV	Amsterdam	dhuisden@roc.nl
K1102	Nitin Bosman	Leidseweg 22	9900BB	Amsterdam	nbosman@roc.nl
K1103	Joseph Demirel	Leidseplein 33	9988BB	Utrecht	josdem@hotmail.com
K1000	Franco Tasiyan	Galileiplantsoen 333	1010RR	Utrecht	frantas@wanadoo.nl
K1901	Akash Kabli	Biohof 55	2299NN	Amstelveen	aka@hetnet.nl
K1902	Tamara Kabli	Mozartstraat 22	3388XX	Amsterdam	tamka@hotmail.com
K1100	Arnold Shaw	Kruislaan 1	9876FF	Rotterdam	asha@roc.nl
K9999	Lex Camilla	Hagabakka 24	4656RR	Utrecht	lecam@wanadoo.nl

6.6.13 De DELETE-opdracht

De DELETE-opdracht verwijdert rijen uit een tabel.

Syntaxis:

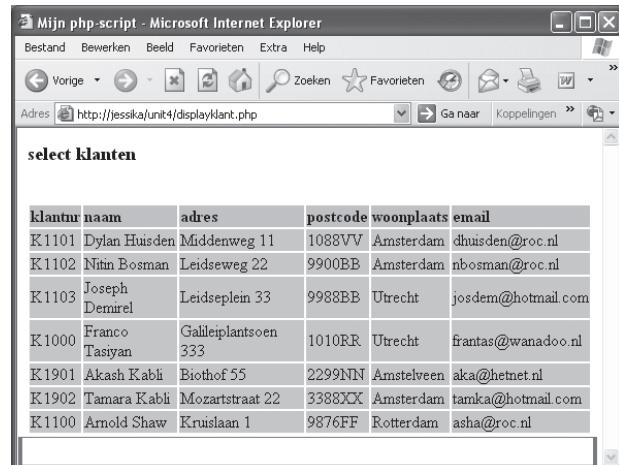
```
DELETE FROM tabel_naam
WHERE voorwaarde
```

Voer de volgende query-opdracht uit:

```
$query = "DELETE FROM klant
WHERE klantnr = 'K9999' ";
```

Figuur 6.66

De resultaatverzameling moet er zo uitzien



klantnr	naam	adres	postcode	woonplaats	email
K1101	Dylan Huisden	Middenweg 11	1088VV	Amsterdam	dhuisden@roc.nl
K1102	Nitin Bosman	Leidseweg 22	9900BB	Amsterdam	nbosman@roc.nl
K1103	Joseph Demirel	Leidseplein 33	9988BB	Utrecht	josdem@hotmail.com
K1000	Franco Tasiyan	Galileiplantsoen 333	1010RR	Utrecht	frantas@wanadoo.nl
K1901	Akash Kabli	Biohof 55	2299NN	Amstelveen	aka@hetnet.nl
K1902	Tamara Kabli	Mozartstraat 22	3388XX	Amsterdam	tamka@hotmail.com
K1100	Arnold Shaw	Kruislaan 1	9876FF	Rotterdam	asha@roc.nl

Maak nu Toets 4, hoofdstuk 6

Maak nu Praktijkopdracht 3, hoofdstuk 6 en Portfolio-opdracht 4 – Orderdatabase en ordertracking

7.1 Inleiding UML

7.1.1 Websoftwareontwikkeling

Een webdatabaseapplicatie is veel meer dan een website met een mailto-formulier. Een webdatabaseapplicatie kan honderden of duizenden regels code hebben.

De volgende technieken kunnen gebruikt worden bij grote webprojecten:

- iteratieve ontwikkeling
- eisenanalyse
- gegevens modelleren
- coderen en testen

Daarnaast zijn er *good practice*-standaarden die zorgen voor de kwaliteit van het ontwikkelingsproces:

- Codeer software die herbruikbaar is.
- Codeer software die eenvoudig te onderhouden is.
- Gebruik een ontwikkelingsplatform.
- Documentatie aanleggen.
- Logica, content en presentatie (PHP, HTML en CSS) apart houden.

7.1.2 Websoftwareanalyse en -ontwerp

Softwareanalyse is een systematische en meetbare benadering voor softwareontwikkeling. Dit is nodig voor complexe maar stabiele websites die goed te onderhouden moeten zijn. Helaas ontbreekt softwareanalyse bij veel webprojecten.

Een reden is dat webontwikkeling vaak documentgeoriënteerd is: documentstructuur, grafisch ontwerp en productie. Deze benadering werkt prima voor kleine en statische sites, maar niet voor complexere sites.

Een tweede reden is dat scriptingtalen, zoals HTML, JavaScript en PHP, gemakkelijk en toegankelijk zijn, zodat men meteen gaat coderen zonder veel aandacht voor analyse.

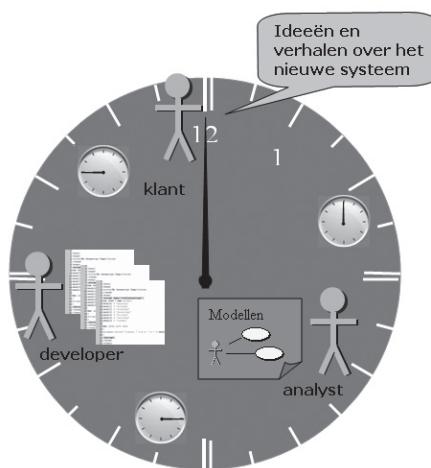
Ten slotte is er bij webprojecten vaak het idee dat er geen tijd is voor enige planning. De resultaten zijn onstabiele applicaties, gemiste deadlines en onleesbare scripts. Het ontwikkelen van webapplicaties is een nieuwe discipline en een methodische benadering is hard nodig.

7.1.3 Iteratieve softwareontwikkeling

Iteratieve softwareontwikkeling is een systematische aanpak voor het plannen en bouwen van webprojecten. Iteratieve softwareontwikkeling zou je als volgt kunnen zien:

Figuur 7.1

Iteratieve
software-
ontwikkeling



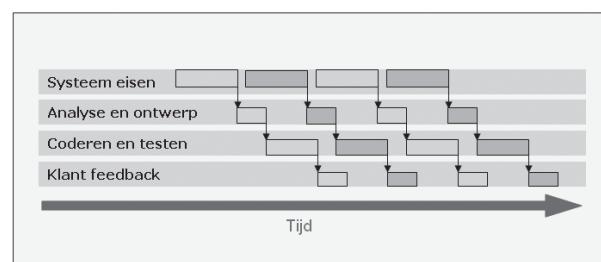
Een iteratie is een cyclus, zoals de wijzer van de klok die ronddraait. Eerst wordt de klant geïnterviewd. Vanuit zijn verhalen en ideeën maak je een eisenanalyse. Daarvoor gebruik je UML-diagrammen. Deze modellen geef je aan de developer. De developer vertaalt je modellen in programmeringcodes. Aan het einde van de eerste iteratie (cyclus) worden de resultaten gepresenteerd aan de klant. De klant geeft vervolgens zijn mening en feedback. Deze feedback is het begin van de tweede iteratie.

7.1.4 Planning softwareontwikkeling

De planning van elke iteratie zou je als volgt kunnen zien:

Figuur 7.2

Planning software-
ontwikkeling



7.2 Modelleren met UML

7.2.1 Wat is UML?

Unified Modeling Language (UML) is een modelleertaal die gebruikmaakt van grafische notatie om modellen te ontwerpen. UML bevordert de communicatie tussen workflowspecialisten, softwaredesigners en andere professionals, zoals mensen met kennis van een sector (bijvoorbeeld verzekeringen, gezondheidszorg of transport).

UML wordt gebruikt voor het maken van modellen voor alle mogelijke systemen:

- Complexe informatiesystemen.
- Technische systemen zoals telecommunicatie.
- Ingebedde realsystemen zoals mobiele telefoons en auto's.
- Softwaresystemen zoals besturingsystemen en databases.
- Bedrijfssystemen zoals beschikbaarheid van mensen en middelen en geldende regels.

Een model is geen definitieve oplossing. Het is maar een van de vele mogelijke oplossingen. Een bruikbaar model moet aan de volgende eisen voldoen:

- *Accuraat*: beschrijft het te bouwen systeem.
- *Consistent*: heeft geen conflicterende informatie.
- *Begrijpelijk*: is eenvoudig uit te leggen en te wijzigen.

7.3 Eisenanalyse met use case-diagrammen

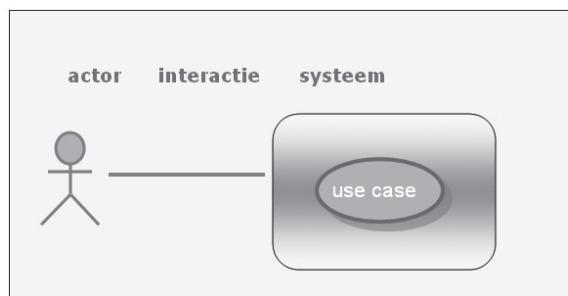
Eisenanalyse zijn de wensen van de opdrachtgever. De ideeën en verhalen van de opdrachtgever vertaal je in een use case-diagram.

7.3.1 Wat is een use case?

Een use case-diagram beschrijft wat een systeem moet doen. Een use case-diagram gebruikt de volgende symbolen:

Figuur 7.3

Een actor in interactie met het systeem

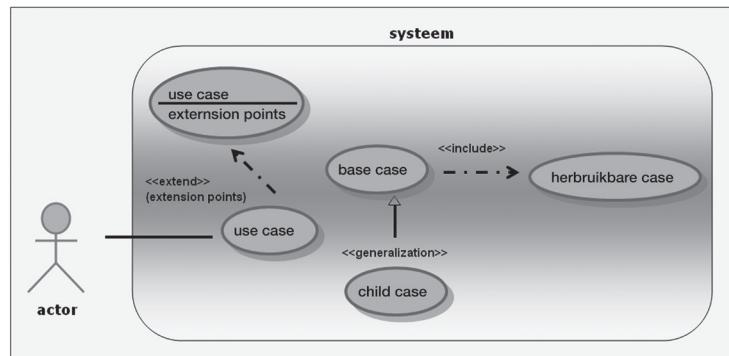


Een actor mag een gebruiker of een ander systeem zijn.

7.3.2 Afspraken over notatie

Use cases gebruiken de volgende notaties om scenario's te modelleren:

Figuur 7.4
Use case-notatie



In figuur 7.4 zie je een acteur in interactie met een systeem. Je ziet ook verschillende relaties ontstaan tussen de use cases.

Een <<include>>- of <<extend>>-relatie geef je aan met de volgende pijl:



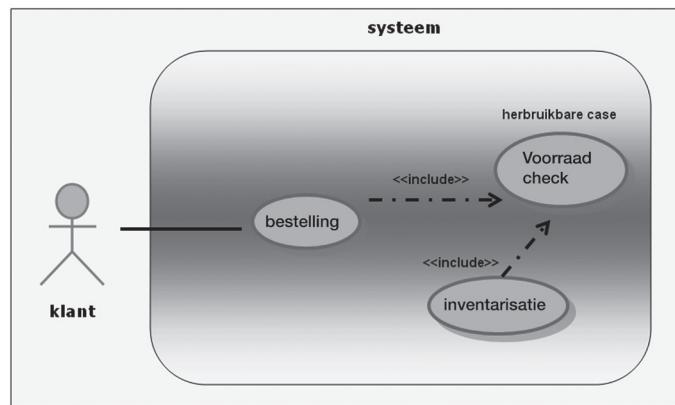
Een <<generalization>>-relatie geef je aan met de volgende pijl:



7.3.3 De <<include>>-relatie

Gebruik <<include>> bij herbruikbare use cases. Bijvoorbeeld, zowel de bestelling als de inventarisatie use cases kunnen gebruikmaken van de voorraad check use case. De voorraad check use case mag gebruikt worden door andere use cases.

Figuur 7.5
Een herbruikbare use case



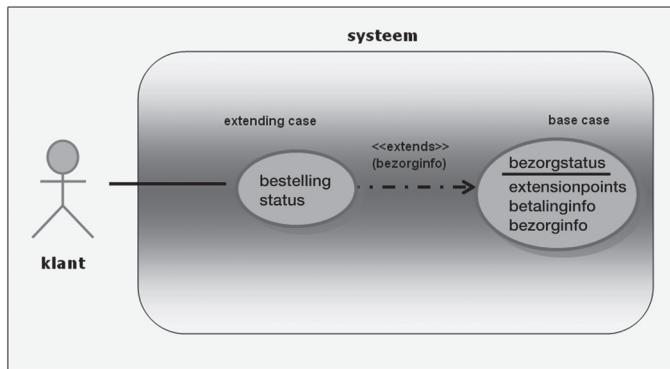
In dit scenario plaatst de klant een bestelling. De bestelling use case maakt een voorraad check. De inventarisatie use case maakt ook gebruik van de voorraad check use case.

Wanneer je dezelfde functionaliteit identificeert binnen twee of meer use cases, kun je een nieuwe use case maken. Zo kunnen meerdere use cases deze functionaliteit hergebruiken.

7.3.4 De <<extend>>-relatie

Gebruik <<extend>> om extension points van de base case toe te voegen aan de extending case. Extension points zijn functionaliteit die je soms gebruikt. Dit is bepaald door de omstandigheden. Het verschil tussen een extend- en een include- relatie is dat een include altijd wordt uitgevoerd.

Figuur 7.6
Een extending
case



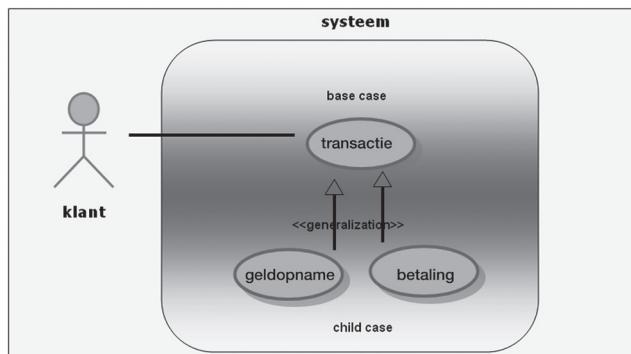
In dit voorbeeld zal de **bestelling status** use case soms (maar niet altijd) **bezorginfo** vragen aan de base case **bezorgstatus**.

Een <<extend>>-relatie gebruik je bij alternatieve functionaliteit. Bijvoorbeeld, onder bepaalde omstandigheden wordt er een alarm geactiveerd.

7.3.5 De <<generalization>>-relatie

Gebruik <<generalization>> bij een use case die gelijksoortig is aan een andere use case, maar ook een alternatief is.

Figuur 7.7
Een generalization
use case

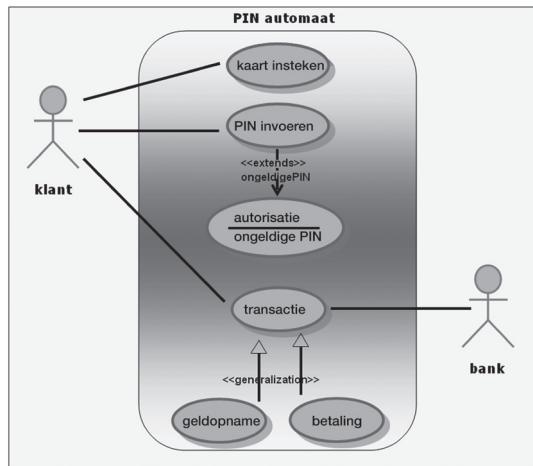


In dit voorbeeld is **transactie** de base use case en **geldopname** en **betaling** zijn de child cases. De child case lijkt erg op de base case, maar is toch een speciale versie van de base case.

7.3.6 Voorbeeld van een geldautomaat use case

Figuur 7.8

Een geldautomaat
use case-diagram



Dit use case-diagram beschrijft het volgende scenario:

1. De klant steekt zijn kaart in.
2. De klant typt zijn pincode in.
3. De geldautomaat voert een pin-autorisatie uit.
4. De klant of de bank kiest een transactie (geldopname of betaling).
5. De geldautomaat voert de transactie uit

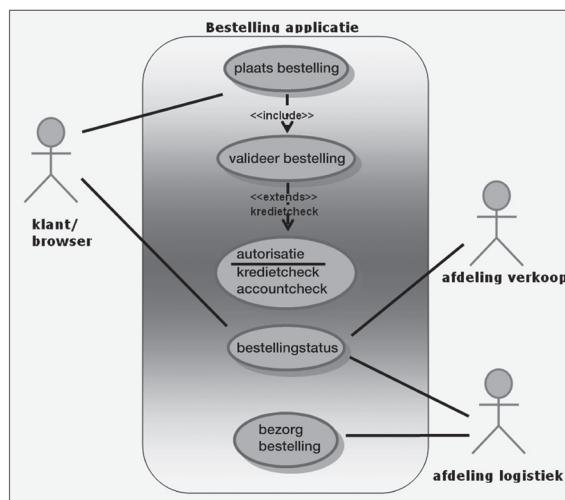
7.3.7 Voorbeeld van een bestelling use case

Dit use case-diagram beschrijft het volgende scenario:

1. Een klant plaatst een bestelling.
2. Het systeem valideert de bestelling.
3. Als het bestelbedrag hoger is dan 1000 euro's, wordt er een krediet check uitgevoerd.
4. De klant, de afdeling verkoop en de afdeling logistiek kunnen de bestelstatus checken.
5. Afdeling logistiek bezorgt de bestelling.

Figuur 7.9

Een bestelling use
case-diagram

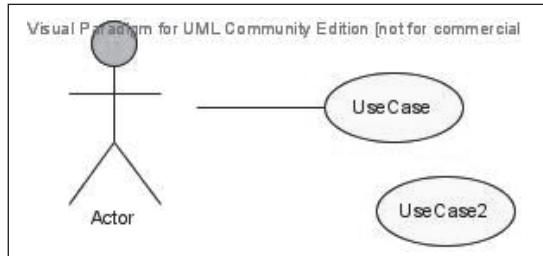


7.3.8 Softwareanalyse met freeware software

Je zou een freeware softwareanalyseprogramma zoals **Visual Paradigm** community edition voor UML kunnen gebruiken. De homepage van Visual Paradigm vind je op: <http://www.visual-paradigm.com/>.

Figuur 7.10

Voorbeeld van een use case gemaakt met Visual Paradigm



Opgave

- 1 Use cases zijn een essentiële tool voor de eisenanalyse en voor de planning en controle van een webproject.

Maak met behulp van visual paradigm een use case-diagram van het volgende bestellingsscenario:

1. De klant plaatst een bestelling.
2. Het systeem valideert de bestelling.
3. Als het bestelbedrag hoger is dan 1000 euro's, wordt een krediet check uitgevoerd.
4. De klant en de afdeling verkoop kunnen de bestellingstatus checken.
5. Afdeling logistiek bezorgt de bestelling.

Voeg de volgende use cases eraan toe:

6. Maak een nieuwe use case **autorisatie** met extension point **ongeldige code**.
7. Maak een nieuwe use case **inloggen** die **ongeldige code** <<extend>>.
8. Maak een nieuwe use case **voorraad check**.
9. Bij de use case **valideer bestelling** <<include>> de nieuwe use case **voorraad check**.

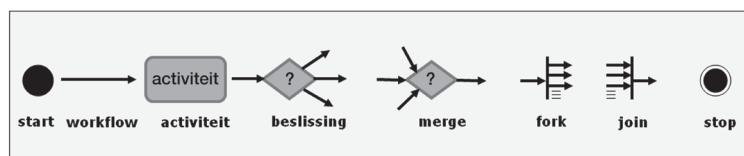
7.4 Van use case naar activiteitendiagram (workflow)

7.4.1 Wat is een activiteitendiagram?

Een activiteitendiagram brengt focus in de stroom van activiteiten van een proces of workflow. Een activiteitendiagram toont hoe activiteiten simultaan lopen of afhankelijk van elkaar zijn. Bij een activiteitendiagram gebruik je de symbolen in figuur 7.11.

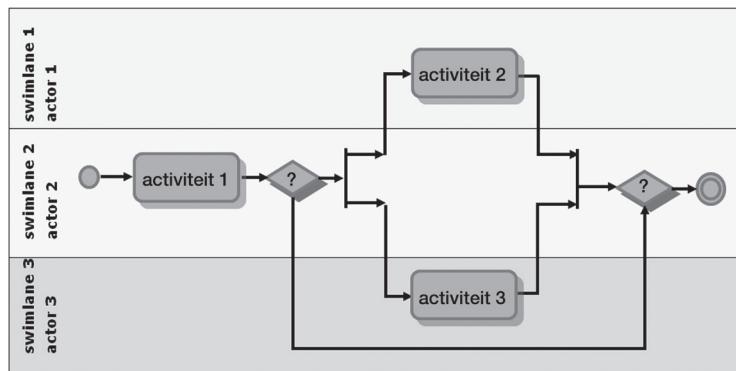
Figuur 7.11

De symbolen van het activiteitendiagram



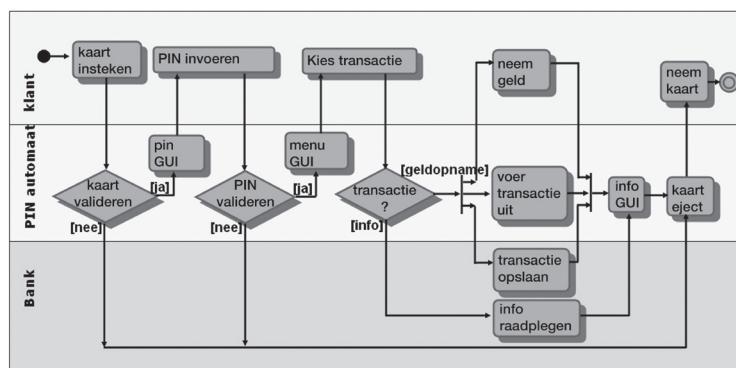
Hierna staat een activiteitendiagram met *swimlanes*. Het diagram geeft weer hoe drie actoren in interactie komen tijdens een workflow of proces. Elke actor krijgt zijn eigen *swimlane*.

Figuur 7.12
Een activiteiten-diagram



Doordat elke actor zijn eigen swimlane krijgt, zijn activiteitendiagrammen ideaal voor het weergeven van processen die parallel plaatsvinden.

Figuur 7.13
Activiteiten-diagram van een geldautomaat



- Opgave 2** Maak met behulp van Visual Paradigm een activiteitendiagram dat gebaseerd is op de use case-scenario van opgave 1.

7.5 Van activiteitendiagram naar sequentiediagram

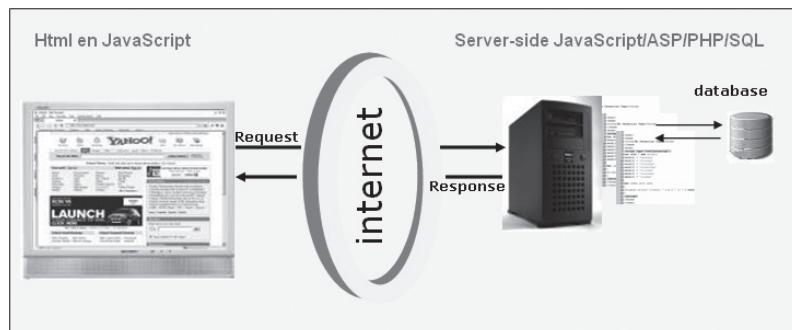
7.5.1 Wat is een sequentiediagram?

Sequentiediagrammen zijn dynamische diagrammen die beschrijven hoe objecten of actoren samenwerken. Een sequentiediagram is een interactiediagram met de details over hoe activiteiten worden uitgevoerd in de tijd. Een sequentiediagram kan maar één use case-scenario tegelijk beschrijven: één pad uit de meerdere paden van een activiteitendiagram.

Sequentiediagrammen zijn zeer effectief voor het beschrijven van tijd gerelateerde zaken, zoals netwerken en interacties tussen use cases.

Figuur 7.14

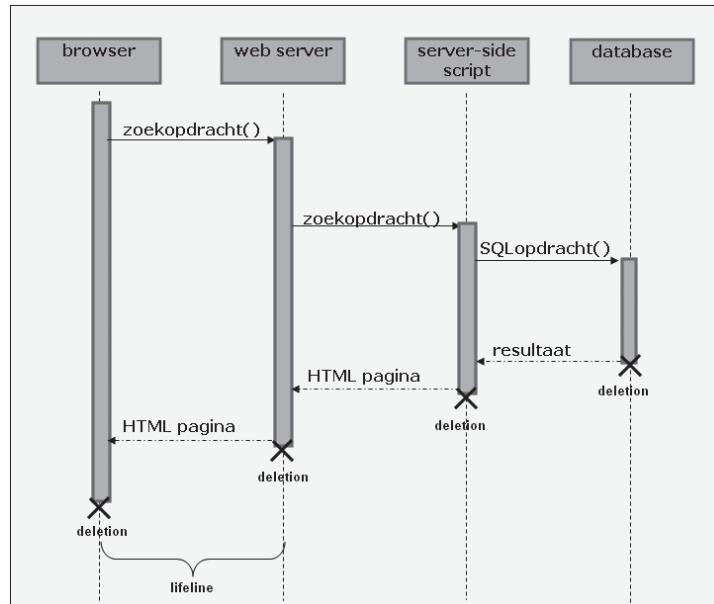
Voorbeeld van een volgorde van evenvents



De activiteiten in figuur 7.14 zie je in het sequentiediagram van figuur 7.15.

Figuur 7.15

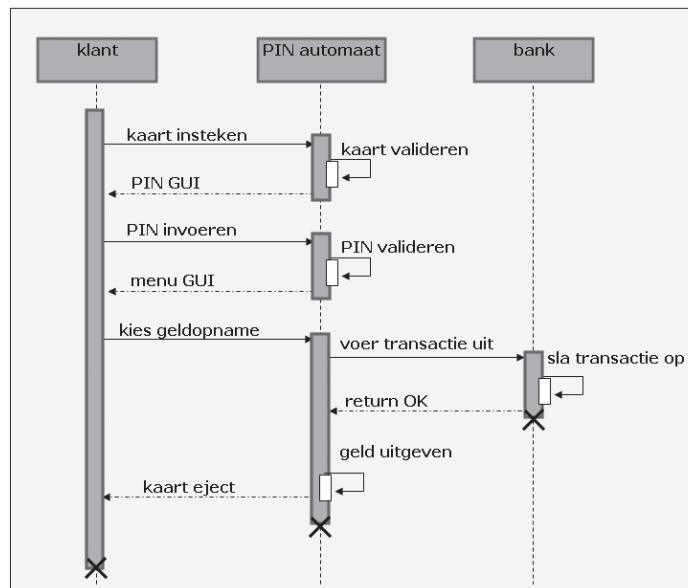
Voorbeeld van een sequentiediagram



De klant stuurt een http-request naar de webserver. De webserver voert het script uit. In het script wordt een SQL-opdracht naar de database verstuurd. Het resultaat wordt in een HTML-pagina weergegeven en de webserver stuurt de HTM-pagina naar de browser.

Figuur 7.16

Voorbeeld van een PIN-automaat sequentie-diagram

**Opgave**

- 3 Maak met behulp van Visual Paradigm een sequentiediagram dat gebaseerd is op het activiteitendiagram van opgave 2.

7.6 Het klassendiagram (class diagram)

Een UML-klassendiagram (class diagram) is een statisch model voor het beschrijven van de objecten in een systeem. Een klasse (class) beschrijft de structuur en de eigenschappen (properties) en tevens de functions (methods) van het object. Ze beschrijft ook de relaties tussen de objecten. We tekenen een klassendiagram als een rechthoek met drie vakken. Een vak voor de naam van de klasse, een vak voor de properties en een vak voor de methods.

Naam
properties
methods

7.6.1 Class met properties

Een class mag properties en methods hebben. Hieronder zie je een klassendiagram voor een class met de naam Persoon en de properties naam, leeftijd en geslacht:

Persoon
naam
leeftijd
geslacht

7.6.2 Class met methods

Een class kan methods hebben. We gebruiken methods meestal om de properties (data) van de class te verwerken. Hieronder is een klassendiagram voor de class Persoon met de method getNaam():



Properties en methods met twee of meer woorden zoals getNaam() schrijven we in camelCase. Dat wil zeggen: zonder spaties en de eerste letter van het tweede woord met een hoofdletter.

7.6.3 Subclass

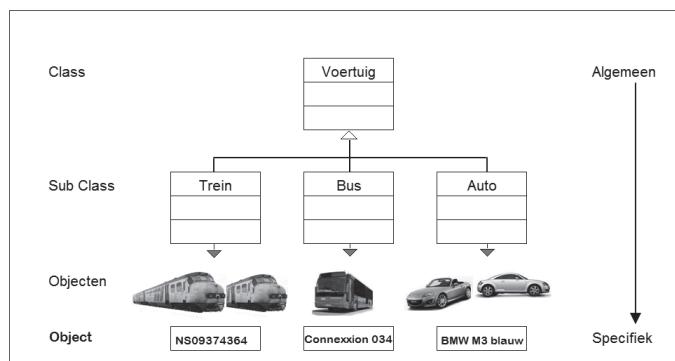
Een class is een algemene beschrijving van een groep objecten. Een subclass is een meer specifieke beschrijving van objecten.

7.6.4 Object

Een object is een voorbeeld van een subclass. Als we eenmaal een subclass gedefinieerd hebben, kunnen we een of meer objecten uit de subclass maken.

De class Voertuig is een algemene beschrijving van voertuigobjecten. De subclass Trein is een specifieker beschrijving van voertuigobjecten. Het object trein NS09374364 is een voorbeeld van de class Voertuig en de subclass Trein.

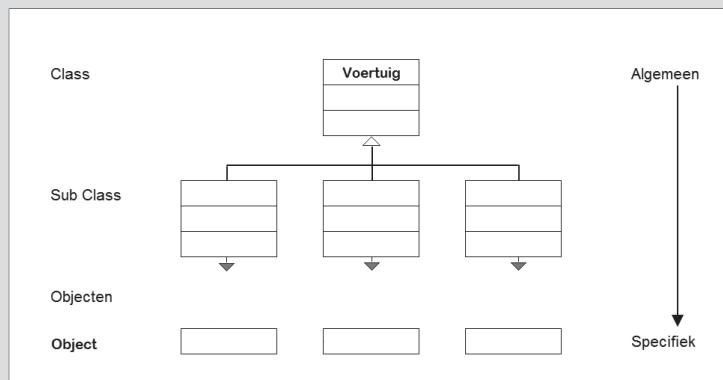
Figuur 7.17
Van algemeen
naar specifiek



- Opgaven** 4 Voor deze opgave bedenk je drie subclasses voor de class Voertuig. Vervolgens bedenk je drie voorbeelden van objecten van de subclasses.

Figuur 7.18

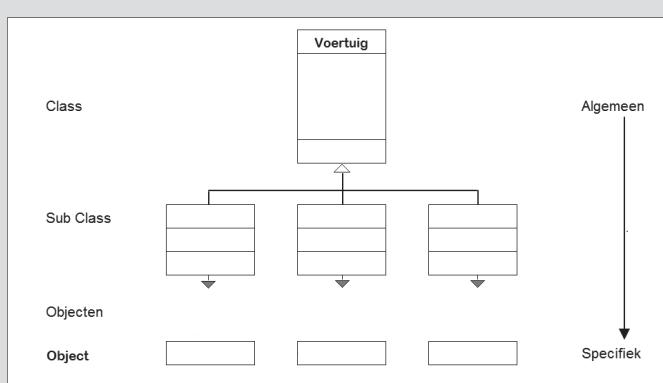
Opgave 4



- 5 Voor deze opgave bedenk je drie properties voor de class Voertuig.

Figuur 7.19

Opgave 5



7.6.5 Relaties

Classes staan niet op zichzelf. Een class heeft relaties met andere classes. Deze relaties zijn *associaties*, *generalisaties* en *afhankelijkheden*.

7.6.6 Associaties

Een associatie is een verbinding tussen twee classes en de afgeleide objecten. De associatie zorgt ervoor dat de objecten '*van elkaar bestaan weten*'. Deze relatie wordt getekend als een doorlopende lijn tussen de twee classes zoals in de volgende figuur:

Figuur 7.20
Associaties

In bovenstaande figuur is de relatie tussen de class Klant en de class Product een associatie. Aan deze associatie hebben we een naam gegeven en deze staat boven de lijn. Deze associatie lezen we als volgt: de Klant bestelt een Product.

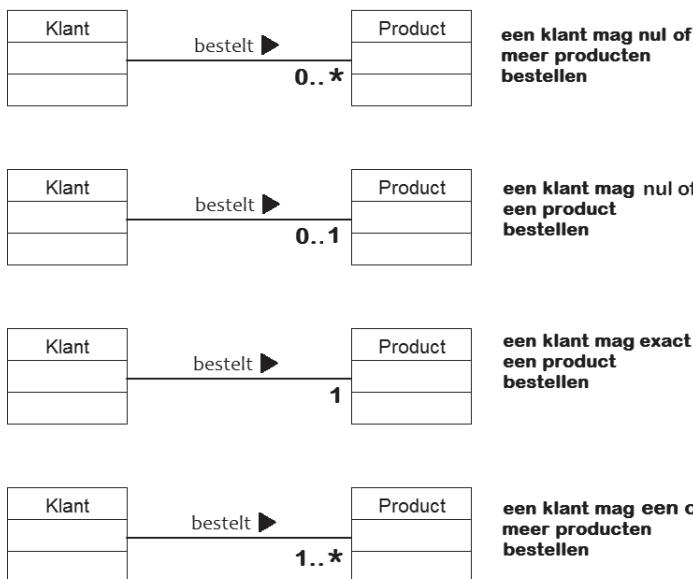
7.6.7 Multiplicity

Een relatie kent multiplicities. Multiplicity betekent ‘veelheid’ en geeft aan hoeveel objecten deelnemen aan de relatie. De volgende tabel geeft de multiplicities en de betekenis aan:

Multiplicity	Betekenis
0	nul
1	een
*	meer
0..*	nul of meer
0..1	nul of een
1..*	een of meer

In de volgende figuur zien we de verschillende mogelijke multiplicities tussen de class Klant en de class Product:

Figuur 7.21
Multiplicity

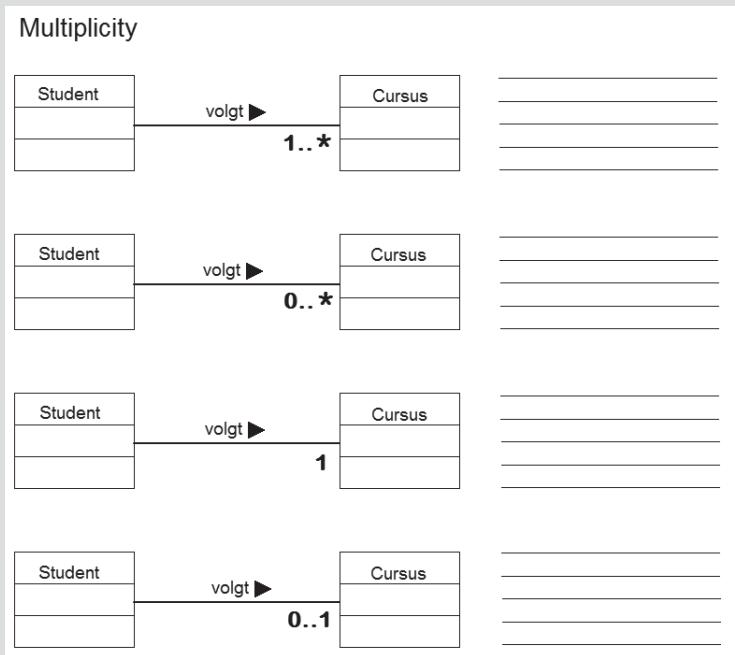


Opgave

- 6 In deze opgave schrijf je aan de rechterkant de betekenis van de aangegeven relatie en multiplicity.

Figuur 7.22

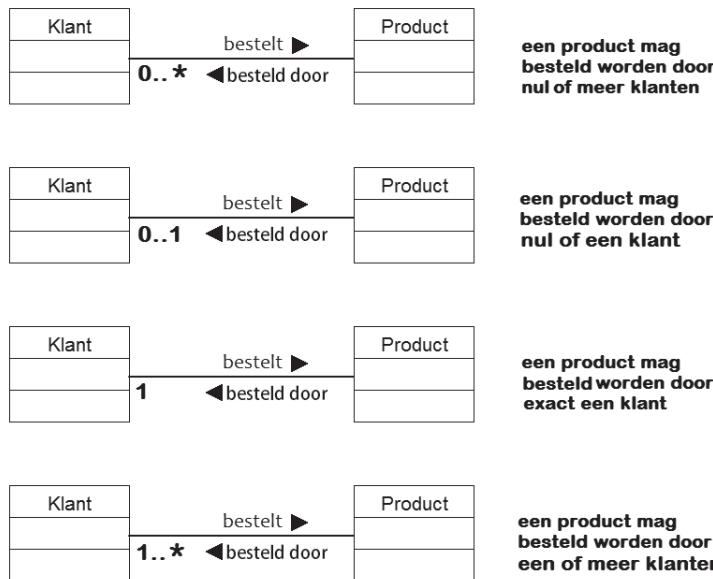
Multiplicity

**7.6.8 Tweerichtingsverkeer relatie**

Een relatie kan ook gelezen worden van rechts naar links. In de volgende figuur krijgt ook de eerste class een multiplicity.

Figuur 7.23

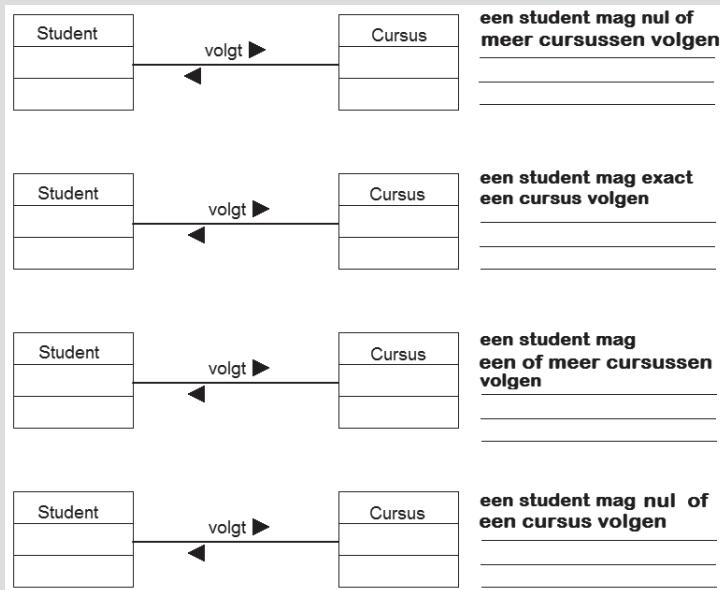
Meerdere multiplicities



- Opgave 7** Geef de tweerichtingsverkeerrelatie aan van de associatie en de multiplicity tussen de klassen Student en Cursus .

Figuur 7.24

Multiplicity:
tweerichtings-
verkeer

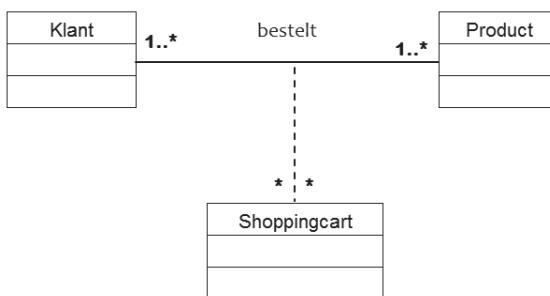


7.6.9 Trio-associatie

Een trio-associatie geeft de relaties aan tussen drie klassen. In de volgende figuur is te zien hoe een klant een of meer producten mag bestellen en hoe elk besteld product geassocieerd is met een shopping cart.

Figuur 7.25

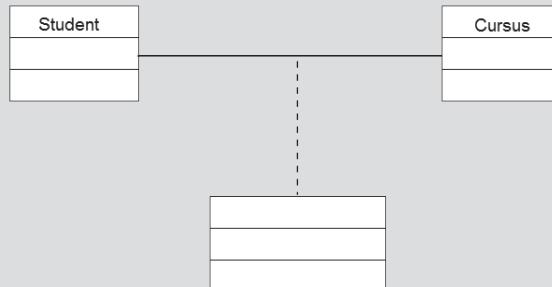
Trio-associatie



- Opgave 8** In deze opgave geef je aan wat associaties en multiplicities tussen de drie classes zijn.

Figuur 7.26

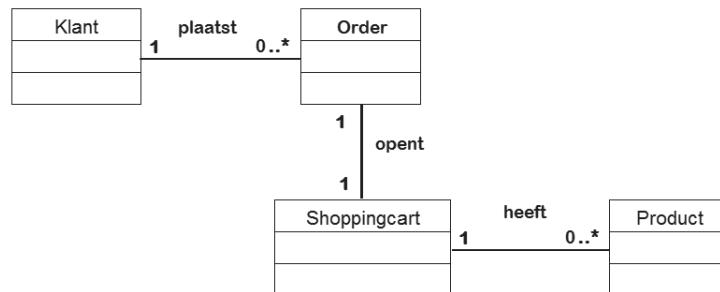
Trio-associatie



In de volgende figuur zien we de relaties tussen de vier classes van een online bestellingsmodel.

Figuur 7.27

Associaties en multiplicities

**Opgave 9**

- Geef hieronder een beschrijving van de relaties en multiplicities van het online bestellingsmodel in de vorige figuur.

Figuur 7.28

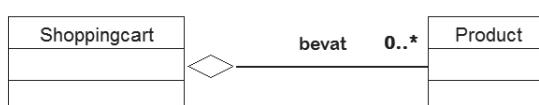
Relaties en multiplicities

7.6.10 Aggregatie

Een aggregatie is een speciale vorm van associatie. Een geaggregeerde class maakt deel uit van een andere class. Een voorbeeld in ons online bestellingsmodel is de class Product.

Figuur 7.29

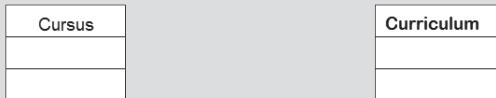
Aggregatie



Deze relatie wordt getekend met een lijn met een lege ruit aan de kant van de class die de andere class bevat. In dit geval bevat de class ShoppingCart de class Product. De class ShoppingCart blijft bestaan, zelfs als er geen Producten zijn.

Opgave 10 Teken de aggregatierelatie tussen de twee classes in de volgende figuur.

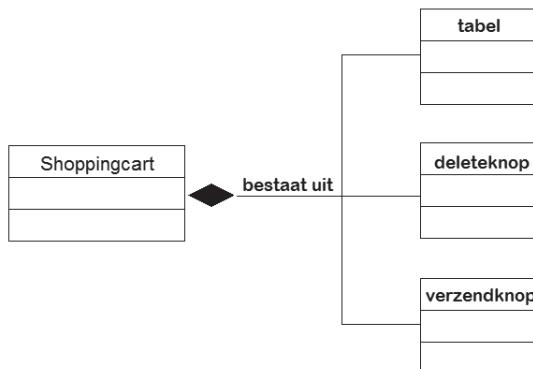
Figuur 7.30
Aggregatierelatie



7.6.11 Compositie-aggregatie

Een compositie-aggregatie is eigenaar van al zijn onderdelen. De eigenaar kan niet bestaan zonder zijn onderdelen. Deze relatie tekenen we met een lijn en een gevulde ruit aan de kant van de eigenaar.

Figuur 7.31
Compositie-
aggregatie



De relatie compositie-aggregatie krijgt de naam “bestaat uit”. De class ShoppingCart kan niet bestaan zonder de compositie van de aggregaties (tabel, deleteknop, verzendknop). Een compositie-aggregatie wordt gecodeerd als een property in de class.



Opgave 11 Teken een compositie-aggregatie-relatie voor een class Bestelformulier.

7.6.12 Encapsulation

In het klassendiagram kunnen we de zichtbaarheid van de properties en methods als public, private en protected bepalen. Dit doen we met de volgende markers.

marker	keyword	betekent
+	public	(publiek) zichtbaar in andere classes
-	private	(privé) zichtbaar alleen in eigen class
#	protected	(beschermd) zichtbaar in eigen class en subclasses

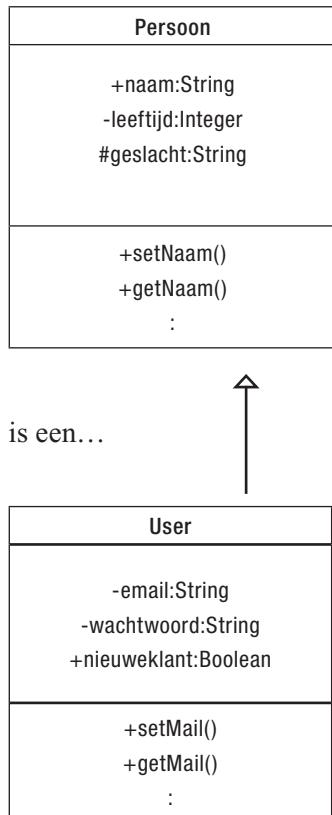
Hieronder geven we de zichtbaarheid aan van de properties in een klassendiagram.



Volgens dit klassendiagram is de property naam public en zichtbaar in alle andere classes. De property geslacht is protected en alleen zichtbaar in deze class en subclasses van deze class. De property leeftijd is private en alleen zichtbaar in deze class.

7.6.13 Generalisatie

De generalisatierelatie is een relatie tussen een algemene class en een specifieke class. De specifieke class bevat aanvullende of specifieke informatie. De specifieke class noemen we een subclass en de algemene class noemen we een superclass. Een generalisatierelatie wordt ook een parent- en child-relatie genoemd. De child class erft alles van de parent class.



De generalisatierelatie tekenen we met een doorlopende lijn vanuit de specifieke naar de algemene class en met een lege driehoek aan de kant van de algemene class. De naam van de relatie beschrijven we als “*is een...*”. In dit geval zeggen we dat de class **User** *is een* class **Persoon** en erft alle properties en methods van de parent class. De class **User** heeft nog meer specifieke properties en methods zoals het e-mailadres, het wachtwoordattribuut en de `setMail()` en `getMail` methods.

- Opgave 12** Teken een generalisatierelatie tussen de class **Persoon** en de subclass **Student**. Bedenk drie properties voor de subclass **Student**.

7.6.14 Abstract class

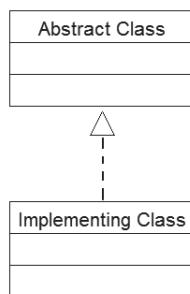
Een abstract class is een class die minstens één abstracte method heeft. Een abstracte method heeft alleen de signatuur van de method en geen body tussen {}, bijvoorbeeld:

```
public function addToCart(Product $product);
```

Een abstract class moet geïmplementeerd worden door een andere class. De implementerende class moet dan de abstracte method implementeren, bijvoorbeeld:

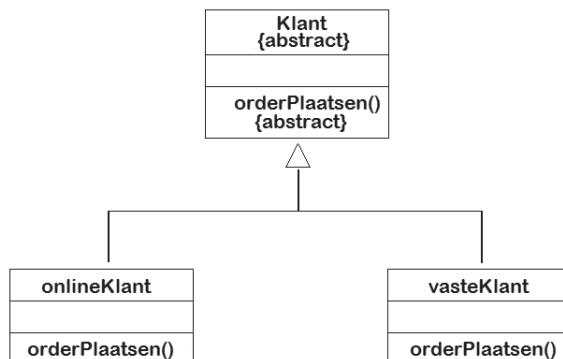
```
public function addToCart( Product $product){
    $this->producten[] = $product;
}
```

Figuur 7.32
Abstract class



Als een class een of meer abstracte methods heeft, is de hele class abstract. Een abstract class krijgt het {abstract}-label naast zijn naam. In de volgende figuur heeft de class Klant de abstract method orderPlaatsen(). De implementerende classes moeten zelf de methode orderPlaatsen() implementeren.

Figuur 7.33
Abstract class

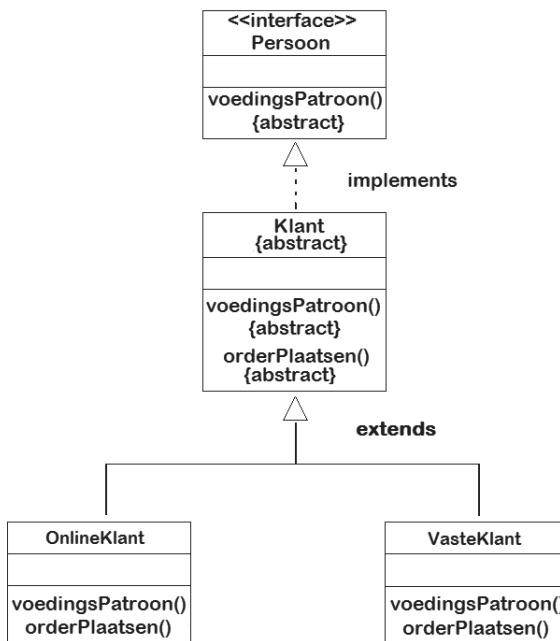


- Opgave 13** Teken een abstract class Voertuig met twee implementerende classes.

7.6.15 Interfaces en Abstract classes

Een interface is in essentie een template (sjabloon). Een interface mag methods declareren maar mag ze niet implementeren. Andere classes mogen de interface implementeren. Een class die een interface implementeert is verantwoordelijk voor het implementeren (coderen) van de methods in de interface.

Figuur 7.34
Interfaces



Maak nu Portfolio-opdracht 5 – Ontwerpen van applicaties

7.7 Object Oriented Programming

Object Oriented Programming (OOP) is een een totaal andere stijl van programmeren. OOP is in de jaren 60 van de vorige eeuw ontwikkeld en tegenwoordig zie je OOP in veel programmeertalen terug, zoals Java en C#. Daarnaast bieden scripttalen zoals PHP, Ruby en Python ook mogelijkheden om OOP toe te passen.

De basiskenmerken van OOP zijn samen te vatten in de volgende termen:

- Classes
- Objects
- Encapsulation
- Inheritance
- Interfaces
- Abstract classes
- Object communication
- Exceptions

Webapplicaties zijn hybride applicaties die geprogrammeerd zijn in HTML5, CSS3, JavaScript, PHP, Ruby, en Pearl. PHP is een hybride programmeertaal met ondersteuning voor Object Oriented Programming en ook functional programming.

7.7.1 Classes

Om een class in PHP te coderen beginnen we met het keyword ‘class’ gevolgd door de naam van de nieuwe class. De naam begint met een hoofdletter. We declareren tegelijkertijd de data van de class met variabelen. Deze variabelen noemen we properties.

Opgave 14 Open je editor en neem de volgende codes over. Sla de codes op als **klassen.php**

```
<?php  
class Persoon {  
    var $naam;  
    var $leeftijd;  
    var $geslacht;  
}  
?>
```

In bovenstaande opgave hebben we de class Persoon gedefinieerd met drie properties: naam, leeftijd en geslacht.

7.7.2 Constructors

Een constructor is een method die automatisch wordt uitgevoerd op het moment dat er een object aangemaakt wordt. De constructor moet ervoor zorgen dat de properties met bepaalde waarden geïnitialiseerd worden. Hiervoor gebruiken we in PHP de `__construct()` method. De construct method begint met twee onderstreepjes (underscore:`_`) gevolgd door het woord *construct*.

Opgave 15 Open **klassen.php** en voeg de volgende constructor eraan toe.

```
<?php

class Persoon{
    var $naam;
    var $leeftijd;
    var $geslacht;
    :

    function __construct($persoonsnaam,
                         $leeftijd,
                         $geslacht){
        $this->naam = $persoonsnaam;
        $this->leeftijd = $leeftijd;
        $this->geslacht = $geslacht;
        echo("<br />Nieuw Persoon object
              $persoonsnaam wordt
              aangemaakt.");
    }
    :
}
?>
```

7.7.3 Objects

We maken nieuw objecten met het keyword *new* gevolgd door de naam van de class van het te maken object. Bijvoorbeeld:

```
$object = new Classnaam();
```

Opgave 16 Open je editor en neem de volgende codes over. Sla ze op als **objecten.php**

```
<html>
<head>
<?php include("klassen.php");?>
<title>OO in PHP</title>
</head>
<body>
<?php
    $umut = new Persoon("Umut",18,"man");
    $demirel = new Persoon("Demirel",23,"man");
?>
</body>
</html>
```

Hier hebben we de twee objecten \$umut en \$demirel met behulp van de constructor gecreëerd. Beide objecten zijn gebaseerd op de class Persoon. De constructor in de class Persoon verwacht drie inputwaarden, namelijk voor naam, leeftijd en geslacht. Bijvoorbeeld:

```
$umut = new Persoon("Umut",18,"man");
```

We zeggen dat het object \$umut een instantie is van de class Persoon. Instantie betekent voorbeeld of exemplaar. We gebruiken altijd het keyword *new* om nieuwe objecten te maken.

Open **objecten.php** met je localhost-server. Het resultaat van de constructors zie je hieronder:

```
Nieuw Persoon-object Umut wordt aangemaakt.  
Nieuw Persoon-object Demirel wordt aangemaakt.
```

7.7.4 Destructor

We hebben gezien hoe de constructor een nieuw object automatisch initialiseert op het moment dat er een nieuw object wordt aangemaakt. Alle objecten hebben een bepaalde levensduur. Garbage collection beheert het computergeheugen door ervoor te zorgen dat alle niet gebruikte objecten automatisch naar de prullenbak worden verplaatst. Met de `__destruct()` method kunnen we onze eigen acties uitvoeren voordat een object automatisch wordt verwijderd.

Opgave 17 Open **klassen.php** en voeg de volgende destructor eraan toe.

```
<?php  
  
class Persoon {  
    public $naam;  
    private $leeftijd;  
    protected $geslacht;  
:  
:  
    function __construct($persoonsnaam,  
                        $leeftijd,  
                        $geslacht){  
        $this->naam = $persoonsnaam;  
        $this->leeftijd = $leeftijd;  
        $this->geslacht = $geslacht;  
        echo("<br />Nieuw Persoon object  
              $persoonsnaam wordt  
              aangemaakt.");  
    }  
  
    function __destruct(){  
        // voer de benodigde acties uit;  
        echo("<br />Persoon object wordt  
              verwijderd");  
    }  
}
```

Aan het einde van je script of op het moment dat er geen verwijzingen meer zijn naar een object wordt het object als vrij gemarkerd door de `__destruct()` method. Alle vrije objecten worden door de garbage collector uit het geheugen verwijderd.

7.7.5 Object verwijderen

We kunnen zelf bepalen wanneer een object verwijderd wordt. Dat doen we met de `unset()` method.

Opgave 18 Open `objecten.php` en voeg de volgende `unset()` method eraan toe.

```
<?php  
:  
:  
    $umut = new Persoon("Umut",18,"man") ;  
    $demirel = new Persoon("Demirel",23,"man") ;  
    unset($umut) ;  
  
?>
```

De `unset()` method roept de `destruct()` method aan. Het resultaat van de `unset()` method zie je hieronder:

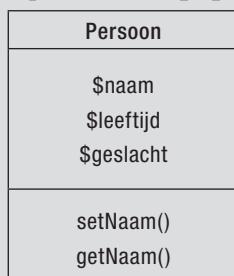
Nieuw Persoon-object Umut wordt aangemaakt.
Nieuw Persoon-object Demirel wordt aangemaakt.
Persoon-object Umut wordt verwijderd.
Persoon-object Demirel wordt verwijderd.

Het object `$umut` hebben we zelf verwijderd met de `unset()` method. Het `$demirel`-object werd automatisch verwijderd aan het einde van het PHP-script.

7.7.6 Access methods

Methods definiëren de functionaliteit van de class. Twee basismethods zijn de `set_` en de `get_` methods. Deze methods gebruiken we om de individuele properties van een class te benaderen.

Opgave 19 Open `klassen.php` en codeer het volgende klassendiagram als volgt:



```
<?php

class Persoon {
    var $naam;
    var $leeftijd;
    var $geslacht;
    :
    :
    :
    function setNaam($persoonsnaam) {
        $this->naam = $persoonsnaam;
    }

    function getNaam() {
        return $this->naam;
    }
}
?>
```

De conventie is om `get_` en `set_` methods te coderen in alle classes. De naam van de methods moet overeenkomen met de naam van de geassocieerde property.

7.7.7 De `$this` pointer

Omdat we de namen van de objecten niet van tevoren kunnen weten, gebruiken we de `$this` pointer. De `$this` pointer is een verwijzing naar de naam van het aangemaakte object. Bijvoorbeeld, als we de volgende `getNaam()` method aanroepen binnen het `$demirel`-object, verwijst `$this` naar het `$demirel`-object.

```
function getNaam() {
    return $this->naam;
}
```

Opgaven 20 Open **klassen.php** en maak `get_` en `set_` methods voor de volgende properties:

```
$leeftijd;
$geslacht;
```

21 Open **objecten.php** en maak het volgende object:

```
naam:Thara
leeftijd:19
geslacht:vrouw
```

Daarna roep je de `setLeeftijd()` method aan en zet je de leeftijd naar 20.

Hoewel de objecten `$demirel` en `$thara` gebaseerd zijn op dezelfde class `Persoon`, zijn zij twee verschillende objecten met eigen properties.

7.7.8 De get_method

We gaan gebruikmaken van de get_method om de waarde van een property op te vragen.

Opgave 22 Open **objecten.php** en voeg de volgende codes eraan toe.

```
<?php
:
:
echo("<br />De leeftijd van Demirel is:" .
    $demirel->getLeeftijd());
echo("<br />De leeftijd van Thara is:" .
    $thara->getLeeftijd());

?>
```

Het resultaat zie je hieronder:

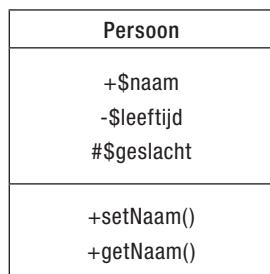
De leeftijd van Demirel is: 23
 De leeftijd van Thara is: 20

7.7.9 Encapsulation

In het klassendiagram kunnen we de zichtbaarheid van de properties en methods als public, private en protected definiëren. Dit doen we met de volgende *markers*.

marker	keyword	betekent
+	public	(publiek) zichtbaar in andere classes
-	private	(privé) zichtbaar alleen in eigen class
#	protected	(beschermd) zichtbaar in eigen class en subclasses

Hieronder geven we de zichtbaarheid aan van de properties in een klassendiagram.



Volgens het klassendiagram is de property \$naam public en zichtbaar in alle andere classes. De property \$geslacht is protected en alleen zichtbaar in deze class en subclasses van deze class. De property \$leeftijd is private en alleen zichtbaar in deze class. In het volgende voorbeeld coderen we de zichtbaarheid volgens bovenstaand klassendiagram.

Opgave 23 Open **klassen.php** en voeg de volgende zichtbaarheid voor de properties en methods eraan toe.

```
<?php

class Persoon {
    public $naam;
    private $leeftijd;
    protected $geslacht;
    :
    :
    :
    :

    public function setNaam($persoonsnaam) {
        $this->naam = $persoonsnaam;
    }

    public function getNaam() {
        return $this->naam;
    }

    :
    :
}

?>
```

7.7.10 Primitieve datatypes

Met de volgende checkmethods kunnen we controleren of de input in de `set_`methods van het juiste datatype is.

method	voorbeeld	resultaat
<code>is_bool()</code>	<code>is_bool(\$naam)</code>	false
<code>is_integer()</code>	<code>is_integer(\$leeftijd)</code>	true
<code>is_double</code>	<code>is_double(\$naam)</code>	false
<code>is_string()</code>	<code>is_string(\$naam)</code>	true
<code>is_object()</code>	<code>is_object(\$demirel)</code>	true
<code>is_array()</code>	<code>is_array(\$geslacht)</code>	false
<code>is_null()</code>	<code>is_null(\$naam)</code>	false

Opgaven 24 Open **klassen.php** en herschrijf de `setNaam()` method met de `is_string()` checkmethod als volgt:

```
function setNaam($persoonsnaam) {
    if(is_string($persoonsnaam))
    {
```

```
    $this->naam = $persoonsnaam;
} else {
    echo("<br />Datatype error in
        setNaam() method.");
}
}
```

- 25** Open **klassen.php** en herschrijf de setLeeftijd() method met de is_integer() checkmethod. Als de input geen integer is, geef je een “Datatype error” aan.
- 26** Open **objecten.php** en voeg de volgende codes eraan toe.

```
<?php
:
:
$thara->setNaam(12343);
$thara->setLeeftijd("twintig");

?>
```

Hier wordt in de setNaam() en de setLeeftijd() methods gecontroleerd op het juiste datatype input. Het resultaat zie je hieronder:

```
Datatype error in setNaam() method.
Datatype error in setLeeftijd() method.
```

7.7.11 Eigen methods

Een class mag zijn eigen methods definiëren. In de volgende opgave coderen we een method die alle properties van de class Persoon weer geeft.

- Opgaven 27** Open **klassen.php** en voeg de volgende method eraan toe.

```
<?php
class Persoon {
    public $naam;
    private $leeftijd;
    protected $geslacht;
:
:
:
    public function printGegevens(){
        $gegevens =
            "<br />De gegevens van $this->naam zijn:
            <br />Leeftijd: $this->leeftijd
            <br />Geslacht: $this->geslacht";
        echo($gegevens);
    }
}
```

- 28** Open **objecten.php** en roep de printGegevens() method van de \$thara- en de \$demirel-objecten aan.

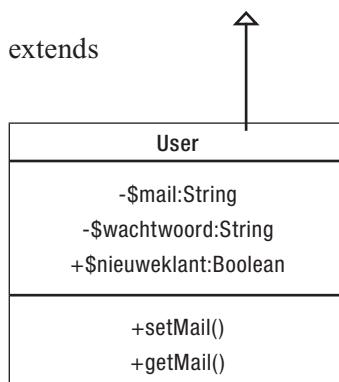
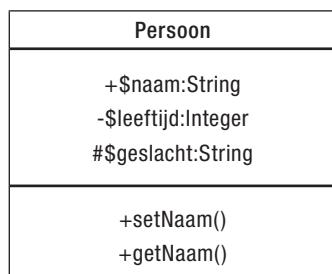
Het resultaat van de printGegevens() method zie je hieronder:

```
De gegevens van Thara zijn:  
Leeftijd: 20  
Geslacht: vrouw  
De gegevens van Demirel zijn:  
Leeftijd: 23  
Geslacht: man
```

7.7.12 Inheritance

Van een superclass of parent-class kunnen we een groep subclasses of child-classes definiëren. De child-classes erven door *inheritance* de public en protected properties en methods van de parent-class. De private properties en methods worden niet overerfd. Een child-class hoeft geen private properties van zijn parent-class te weten.

In het volgende klassendiagram zie je dat de class User een child-class is van de class Persoon. We zeggen dat de class User de class Persoon *extends*.



De class User erft de properties \$naam en \$geslacht en ook de get_ en set_ methods uit de class Persoon. De class User bevat ook zijn eigen properties \$mail, \$wachtwoord en \$nieuweklant en get_ set_ methods.

7.7.13 Constructors en inheritance

In PHP maken we nieuwe subclasses met het keyword "extends". In de volgende opgave coderen we bovenstaand klassendiagram.

Opgave 29 Open **klassen.php** en voeg de volgende subclass eraan toe.

```
<?php
:
:
class User extends Persoon {
    private $mail;
    private $wachtwoord;
    public $nieuweklant;

    function __construct(
        $persoonsnaam,$leeftijd,
        $geslacht,$mail,
        $wachtwoord,$nieuweklant){

        parent::__construct($persoonsnaam,
                            $leeftijd,
                            $geslacht);
        $this->mail = $mail;
        $this->wachtwoord = $wachtwoord;
        $this->nieuweklant = $nieuweklant;
        echo("<br />Nieuw User object
              $persoonsnaam wordt aangemaakt");
    }

    function __destruct(){
        // voer de benodigde acties uit;
        echo("<br />Nieuw User extends Persoon");
    }

    public function setMail($mail){
        $this->mail = $mail;
    }
    public function setWachtwoord($wachtwoord){
        $this->wachtwoord = $wachtwoord;
    }
    public function setNieuweklant($nieuweklant){
        $this->nieuweklant = $nieuweklant;
    }

    public function getEmail(){
        return $this->email;
    }
}
```

```

        public function getWachtwoord(){
            return $this->wachtwoord;
        }
        public function getNieuweklant(){
            return $this->nieuweklant;
        }

    }

```

De constructor maakt als volgt gebruik van de constructmethode van de parent-class:

```

parent::__construct($persoonsnaam,
                    $leeftijd,
                    $geslacht);

```

Om een methode van een superclass aan te roepen gebruiken we het keyword **parent::**

7.7.14 Inheritance implementeren

We moeten voorzichtig zijn wanneer we een object van een child-class maken.

Het volgende is bijvoorbeeld fout:

```
$user1=new User("Shireen@gmail.com","Uh65Tg",true);
```

Het is belangrijk dat we ook de waarden van de overerfde properties aan de constructor doorgeven. Anders is het gemaakte object niet compleet.

Het volgende is correct:

```
$user1 = new User("Shireen ",22,"vrouw",
                  "Shireen@gmail.com","Uh65Tg",true);
```

Opgave 30 Open **objecten.php** en voeg het volgende object eraan toe.

```

<?php
:

$user1 = new User("Shireen ",22,"vrouw",
                  "Shireen@gmail.com","Uh65Tg",true);
echo("<br />Naam van user1 is:" .
     $user1->getNaam() );
?>

```

Het resultaat van de echo-opdracht zie je hieronder:

Nieuw Persoon-object Shireen wordt aangemaakt
Nieuw User extends Persoon
Naam van user is: Shireen

7.7.15 Overschrijven parent methods

De getNaam() method was overerfd van de class Persoon naar de class User. Dus een user1-object kon deze method uitvoeren zoals in de vorige opgave.

```
$user1->getNaam();
```

Maar kijk wat er gebeurt met de overerfde printGegevens() method.

Opgave 31 Open **objecten.php** en voeg het volgende object eraan toe.

```
<?php  
:  
    $user1->printGegevens();  
?>
```

Het resultaat van de printGegevens() method zie je hieronder:

```
De gegevens van Shireen zijn:  
Leeftijd: 22  
Geslacht: vrouw
```

De gegevens van de overerfde printGegevens() method in het object user1 zijn niet compleet. Dit betekent dat de child-class de printGegevens() method van de parent-class moet overschrijven met zijn eigen printGegevens() method.

Opgave 32 Open **klassen.php** en voeg de volgende method eraan toe.

```
<?php  
:  
:  
class User extends Persoon {  
    private $mail;  
    private $wachtwoord;  
    public $nieuweklant;  
:  
:  
    public function printGegevens(){  
        $gegevens = parent::printGegevens();  
        $gegevens .= "<br />Email: $this->mail";  
        $gegevens .= "<br />Wachtwoord:  
                    $this->wachtwoord";  
        $gegevens .= "<br />Nieuweklant:  
                    $this->nieuweklant";  
        echo($gegevens);  
    }  
}  
?>
```

De overgeschreven printGegevens() method voert eerst de parent-class printGegevens() method als volgt uit:

```
$gegevens = parent::printGegevens();
```

Daarna worden de gegevens van de child-class toegevoegd. Het resultaat van de overschreven printGegevens() method zie je hieronder:

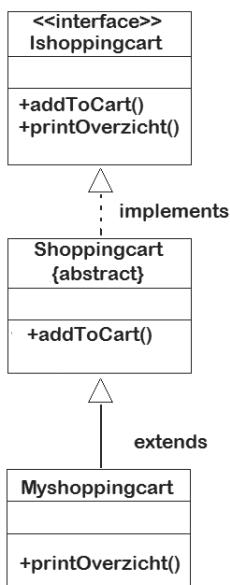
De gegevens van Shireen zijn:

Leeftijd: 22
Geslacht: vrouw
Email: Shireen@gmail.com
Wachtwoord: Uh65Tg
Nieuwklant: 1

7.7.16 Interfaces

Een interface is een template. Een interface mag methods declareren maar mag ze niet implementeren. Andere classes mogen de interface implementeren. Een class die een interface implementeert is verantwoordelijk voor het implementeren (coderen) van de methods die genoemd staan in de interface.

Figuur 7.35
Klassendiagram
19



In dit klassendiagram is de Ishoppingcart interface niet meer dan een template voor ShoppingCart-classes. Het benoemt de methods die een ShoppingCart-class moet hebben. De ShoppingCart-class implementeert (codeert) de addToCart() method. De Myshoppingcart-class extends de ShoppingCart-class en implementeert (codeert) de printOverzicht() method. Bovenstaande klassendiagram coderen we als volgt: eerst coderen we de Ishoppingcart interface.

Opgave 33 Open **klassen.php** en voeg de volgende class eraan toe.

```
interface Ishoppingkart{
    public function addToCart();
    public function printOverzicht();
}
```

7.7.17 Abstract class

Vervolgens coderen we de abstract class ShoppingCart. Deze class implementeert de Ishoppingcart interface en implementeert de addToCart() method. De class ShoppingCart blijft een abstract class want de method printOverzicht() wordt niet gecodeerd.

Opgave 34 Open **klassen.php** en voeg de volgende class eraan toe.

```
abstract class ShoppingCart implements Ishoppingcart {
    protected $producten = array();

    public function addToCart( Product $product){
        $this->producten[] = $product;
    }

}
```

Volgens het klassendiagram coderen we nu de class MyShoppingcart. Deze class extends de ShoppingCart-class en erft de addToCart() method. De class implementeert de printOverzicht() method.

Opgave 35 Open **klassen.php** en voeg de volgende class eraan toe.

```
class MyShoppingcart extends ShoppingCart {
    public function printOverzicht(){
        echo("<table border=1>

```

```
$rij .= "<td>".$product->getModel()."</td>";
$rij .= "<td>".$product->getPrijs()."</td>";
$rij .= "<td>".$product->getAantal()."</td>";
$rij .= "<td>".$product->getKorting()."</td>";
$rij .= "</tr>";
echo($rij);
}
echo("</table>");
}
?>
```

In de volgende opgave coderen we een class Product met productinformatie over laptops.

Opgave 36 Open **klassen.php** en voeg de volgende class eraan toe.

```
class Product {
    private $id;
    private $beschrijving;
    private $merk;
    private $model;
    private $prijs;
    private $aantal;
    private $korting;

    function __construct($id,
                        $beschrijving,
                        $merk,
                        $model,
                        $prijs,
                        $aantal,
                        $korting){
        $this->id = $id;
        $this->beschrijving = $beschrijving;
        $this->merk = $merk;
        $this->model = $model;
        $this->prijs = $prijs;
        $this->aantal = $aantal;
        $this->korting = $korting;
        echo("<br />Nieuw Product object
             beschrijving wordt aangemaakt");
    }

    function __destruct(){
        // voer de benodigde acties uit;
        echo("<br />Product object
             $this->beschrijving wordt verwijderd");
    }

    public function setId($id){
        $this->id = $id;
    }
```

```
public function setBeschrijving($beschrijving){
    $this->beschrijving = $beschrijving;
}
public function setMerk($merk){
    $this->merk = $merk;
}
public function setModel($model){
    $this->model = $model;
}
public function setPrijs($prijs){
    $this->prijs = $prijs;
}
public function setAantal($aantal){
    $this->aantal = $aantal;
}
public function setKorting($korting){
    $this->korting = $korting;
}

public function getId(){
    return $this->id;
}
public function getBeschrijving(){
    return $this->beschrijving;
}
public function getMerk(){
    return $this->merk;
}
public function getModel(){
    return $this->model;
}
public function getPrijs(){
    return $this->prijs;
}
public function getAantal(){
    return $this->aantal;
}
public function getKorting(){
    return $this->korting;
}

public function printProductInfo(){
    $rij = "<tr><td>$this->id</td>";
    $rij .= "<td>$this->beschrijving</td>";
    $rij .= "<tr><td>$this->merk</td>";
    $rij .= "<tr><td>$this->model</td>";
    $rij .= "<tr><td>$this->prijs</td>";
    $rij .= "<tr><td>$this->aantal</td>";
    $rij .= "<tr><td>$this->korting</td></tr>";
    echo($rij);
}
}
```

7.7.18 Object Communication

Objecten communiceren onderling via methods, bijvoorbeeld wanneer een object een method aanroeft vanuit een ander object. In de volgende opgave maken we twee productobjecten en een shoppingcart-object.

Vanuit het shoppingcart-object voegen we de twee productobjecten aan de shoppingcart toe. Vervolgens voeren we de printOverzicht() method uit.

Opgave 37 Open **objecten.php** en voeg de volgende objecten eraan toe.

```
$product1 = new Product("001",
    "laptop",
    "Toshiba",
    "Satelite",
    999.99,
    10,
    true);
$product1->printProductInfo();
$product2 = new Product("002",
    "laptop",
    "Acer",
    "Aapire",
    1099.99,
    5,
    true);
$product2->printProductInfo();
$myShoppingcart = new MyShoppingcart();
$myShoppingcart->addToCart($product1);
$myShoppingcart->addToCart($product2);
$myShoppingcart->printOverzicht();
```

Het resultaat moet er als volgt uitzien:

Nieuw Productobject laptop wordt aangemaakt001laptopToshibaSatelite999.99101

Nieuw Productobject laptop wordt aangemaakt002laptopAcerAapire1099.9951)

Bestellingoverzicht

Product ID	Beschrijving	Merk	Model	Prijs	Aantal	Korting
001	laptop	Toshiba	Satelite	999.99	10	1
002	laptop	Acer	Aapire	1099.99	5	1

7.7.19 Foutafhandeling met Exceptions

Een exception is een speciaal object uit de class Exception. Exception-objecten zijn ontworpen om foutmeldingen te rapporteren. De constructor van de class Exception vraagt twee optionele parameters, een foutmelding-string en een foutcode. De class bevat de volgende methods:

Method	Resultaat
getMessage()	De foutmelding van de Exception
getCode()	De foutcode van de Exception
getFile()	De bestandsnaam die de Exception genereerde
getLine()	Het regelnummer dat de Exception genereerde
getPrevious()	Een genest exception-object
getTrace()	Een multidimensionale array met method calls die de Exception genereerde
getTraceAsString()	Een string met de method calls die de Exception genereerde.
__toString()	

7.7.20 Exceptions werpen

Het ‘werpen’ van een Exception doen we met het keyword *throw* en het object Exception. Na het werpen van een Exception-object wordt de uitvoering van de huidige code gestopt en wordt de programmacontrole teruggegeven aan de aanroepende code waar de Exception afgehandeld moet worden. Een voorbeeld van een Exception is wanneer een bestand niet kan gevonden worden.

- Opgaven 38** Open **klassen.php** en herschrijf de setAantal() method van de class Product met de `is_integer()` check method als volgt:

```
public function setAantal($aantal){
    if(is_integer($aantal))
    {
        $this->aantal = $aantal;
    } else {
        throw new Exception("<br />Datatype
                           Exception in setAantal() method.");
    }
}
```

- 39** Open **klassen.php** en herschrijf de setLeeftijd() method van de class Persoon class.

7.7.21 De try-catch clause

Als je een method aanroeft die een Exception zou kunnen werpen, moet je de aanroepende code binnen een try-catch clause coderen.

Opgave 40 Open **objecten.php** en voeg de volgende codes eraan toe.

```
<?php
:
:
try{
    $product1->setAantal("drie");
} catch (Exception $e){
    die( $e-> __toString());
}
?>
```

Hier wordt de setAantal() method gecontroleerd op het juiste type data-input. Als de input een Datatype Exception is dan stopt de catch-clausule het script en geeft de Exception weer. Het resultaat zie je hieronder:

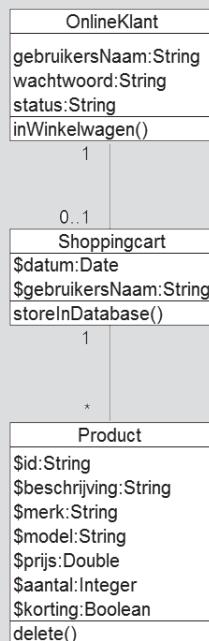
```
Datatype Exception in setAantal() method.' in C:\XAMP\htdocs\OOP\klassen.
php:136 Stack trace: #0 C:\XAMP\htdocs\OOP\objecten.php(43): Product-
>setAantal('drie') #1 {main}
```

UML code camp - Lab 1

Codeer de classes uit onderstaand klassendiagram.

Figuur 7.36

Klassendiagram



Voorkennis

Kennis van en ervaring met webapplicatieontwikkeling met name HTML en JavaScript is vereist. Evenals kennis van een in een serveromgeving gebruikte taal (zoals PHP) en een DBMS (zoals MySQL).

8.1 Inleiding XML

XML (een soort afkorting voor Extensible Markup Language) is een open standaard voor het coderen van elektronische documenten, ontwikkeld door organisatie W3C. XML is een ‘meta-taal’ voor het definiëren van nieuwe talen zoals XHTML, SOAP en RSS, en voor kantoorapplicaties zoals OpenOffice. XML was ontworpen voor het opslaan, delen en transporteren van data.

Verschil tussen XML en HTML

XML gebruiken we voor het *structureren* en *transporteren* van informatie. We gebruiken XHTML voor het *weergeven* van informatie. XML heeft geen voorgedefinieerde tags. Je moet je eigen tags definiëren. XHTML heeft een set van gedefinieerde tags.

Verschil tussen XML en databases

XML is geen relationele database (RDBMS). We kunnen datastructuren in XML definiëren, opslaan en transporteren. We kunnen zelfs XML-queries uitvoeren. Maar we kunnen geen relaties tussen de datastructuren leggen, zoals in een relationele database (bv. in MySQL) het geval is. De data in een RDBMS is in binair formaat en dus veel sneller te verwerken dan het tekstformaat van XML.

Net als alle andere datastructuren doet XML zelf eigenlijk niks. Als je gegevens uit een DBMS wilt verwerken dan doe je dat met SQL. Als je gegevens uit datastructuren in XML wilt verwerken dan moet je dat doen met behulp van andere software, zoals JavaScript of PHP.

In dit hoofdstuk kijken we eerst naar XML-datastructuren en schema's. Daarna gaan we verder met het combineren van XML met PHP en

MySQL. Daarna kijken we naar de combinatie van XML + JavaScript + PHP = AJAX. Deze combinatie levert veel mogelijkheden voor het coderen van interactieve applicaties.

Waarvoor gebruiken we XML?

XML is ideaal voor het structureren, transporteren en delen van data.

XML kan dus een interface zijn voor de transmissie van data tussen twee verschillende systemen. Het is bijvoorbeeld veel complexer om een Oracle-database rechtstreeks te laten communiceren met een DB2- of MySQL-database. Inmiddels zijn er al diverse technologieën die gebaseerd zijn op XML:

- XML schema's voor het definieren van XML-structuren en datatypes
- XSLT om XML-data te transformeren
- SOAP voor het uitwisselen van XML-data tussen applicaties
- WSDL om webservices te definiëren
- XPath en XQuery om XML-data te benaderen
- SMIL voor het definiëren van graphics

8.1.1 XML-documenten

XML-documenten kunnen we met kladblok, XML-editors of HTML-editors maken. Aan de hand van het volgende voorbeeld gaan we oefenen met XML. De tags die we daar gebruiken zijn geen pregedefinieerde tags, zoals <p> of <h1> in HTML. In XML moeten we de tags zelf definiëren. Door het definieren van eigen tags bouwen we een ‘selfdescriptive’ datastructuur op, een datastructuur die zichzelf beschrijft.

Opgave 1 Open je editor en typ het volgende XML-bestand over. Sla het bestand op als **nieuwetitels.XML**.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<nieuwetitels>
    <leverancier>
        <naam>Boeken-Plus</naam>
        <code>10394</code>
        <datum>2010-10-10</datum>
    </leverancier>
</nieuwetitels>
```

Het root-element

Een XML-document moet altijd beginnen met het root-element. In bovenstaand document is <nieuwetitels> het root-element. Een XML-document mag maar één root-element hebben. Het root-element sluit je als laatste af</nieuwetitels>

Elementen

Na het root-element definieer je je elementen. In bovenstaand document hebben we het element *leverancier* met de drie subelementen *naam*, *code* en *datum* gedefinieerd. In de volgende opgave voegen we het boek-element eraan toe.

Opgave 2 Open **nieuwetitels.XML** en voeg het volgende boek-element eraan toe.

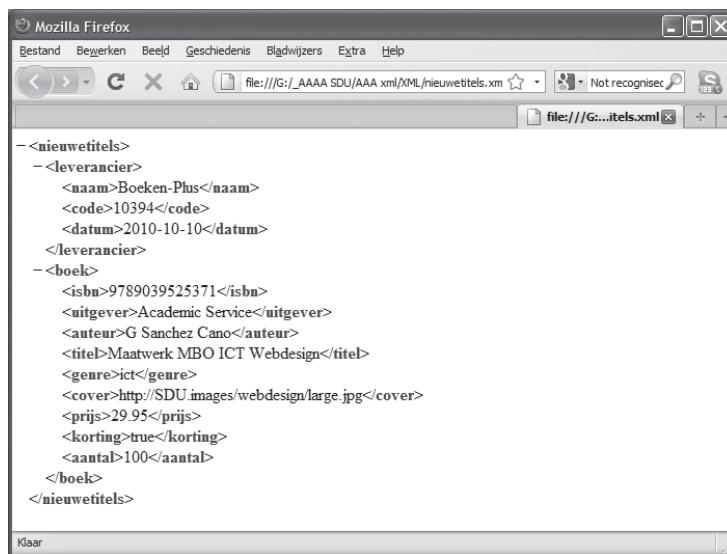
```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<nieuwetitels>
    <leverancier>
        <naam>Boeken-Plus</naam>
        <code>10394</code>
        <date>2010-10-10</date>
    </leverancier>

    <boek>
        <isbn>9789039525371</isbn>
        <uitgever>Academic Service</uitgever>
        <auteur>G Sanchez Cano</auteur>
        <titel>Maatwerk MBO ICT Webdesign</titel>
        <genre>ict</genre>
        <cover>http://SDU.images/webdesign/large.jpg</cover>
        <prijs>39.95</prijs>
        <korting>true</korting>
        <aantal>100</aantal>
    </boek>
    <boek>
        .
        .
        .
    </boek>
</nieuwetitels>
```

Een XML-document is geen XHTML-document. Als je **nieuwetitels.XML** opent met je browser dan zie je de structuur zoals weergegeven in figuur 8.1.

Figuur 8.1

Een XML-document geopend met een webbrowser

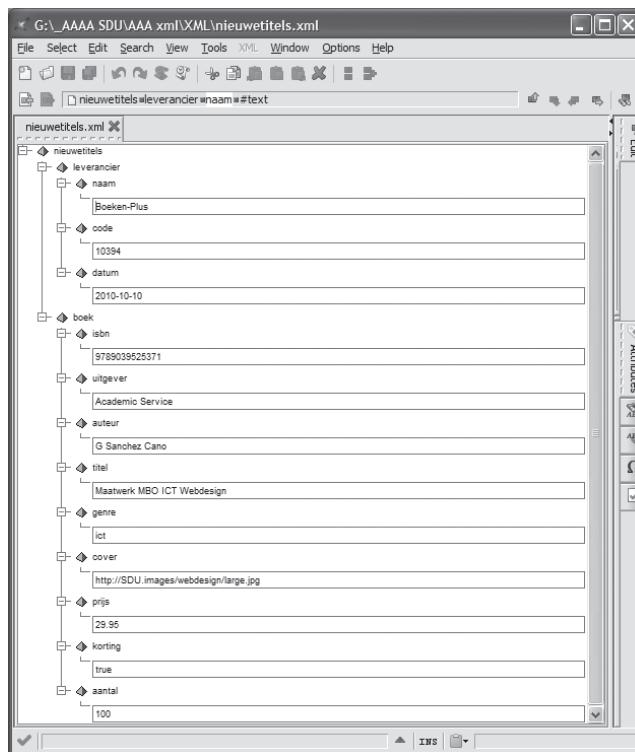


XML-editors

XML-editors zijn gemaakt om je te helpen met het maken van correcte XML-documenten. Er zijn verschillende opensource XML-editors op

internet. In figuur 8.2 is het bestand **nieuwetitels.XML** geopend met een XML-editor.

Figuur 8.2
Een XML-document geopend met een XML-editor



Attributen

Elementen in een XML-structuur mogen attributen hebben. De waarde van de attributen geven we tussen aanhalingstekens aan. Bijvoorbeeld, het element <titel> zou ook het attribuut genre kunnen hebben als volgt:

```
<titel genre="ict"> Maatwerk MBO ICT Webdesign</titel>
```

Maar in onze structuur hebben we <genre> aangegeven als een element van de structuur. Omdat elementen veel makelijker te verwerken zijn gebruiken we geen attributen.

Uitbreiden van XML-elementen

XML-elementen zijn ‘extensible’, ofwel uitbreidbaar. Als we later onze datastructuur verder uitbreiden, heeft dat geen gevolgen voor onze applicaties. De toegevoegde informatie die bedoeld is voor nieuwe applicaties heeft geen consequenties voor de oude applicaties.

XML tree

Onze datastructuur heeft de vorm van een boom; vaak wordt ook het Engelse woord ‘tree’ gebruikt. Deze structuur begint met het root-element en eindigt met de subelementen. De volgende code geeft een voorbeeld van een simpele datastructuur:

```

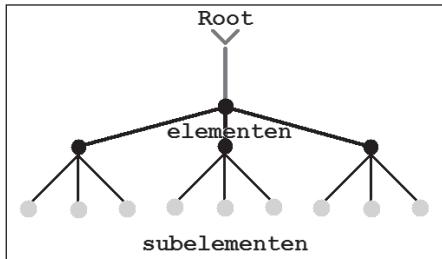
<root>
  <element>
    <subelement>content</subelement>
    <subelement>content</subelement>
    <subelement>content</subelement>
  </element>
  <element>
    <subelement>content</subelement>
    <subelement>content</subelement>
    <subelement>content</subelement>
  </element>
  <element>
    <subelement>content</subelement>
    <subelement>content</subelement>
    <subelement>content</subelement>
  </element>
</root>

```

Deze datastructuur heeft een root en drie elementen met elk drie sub-elementen. Dit kunnen we ook in een figuur weergeven als een soort (omgekeerde) boom.

Figuur 8.3

Een eenvoudige boomstructuur



Wel gevormde XML-documenten

Een XML-document kun je helemaal zelf opbouwen, maar moet wel aan een aantal regels voldoen. Een document dat aan de regels voldoet, noemen we *wel gevormd*. Een wel gevormd XML-document voldoet aan de volgende eisen:

- elke begintag (bv. <tagnaam>) moet een bijgehorende eindtag hebben (bv. </tagnaam>);
- tags mogen ‘genest’ worden;
- tags mogen niet overlappen;
- namen van elementen worden in kleine letters gegeven;
- namen mogen letters, cijfers en andere tekens bevatten;
- namen mogen niet beginnen met een cijfer;
- namen mogen niet beginnen met ‘XML’;
- namen mogen geen spaties bevatten;

Code	Opmerking
<2de naam>Franklin</2de naam>	Fout: de naam begint met een cijfer en bevat een spatie

Niet alleen tagnamen van een element moeten aan eisen voldoen, ook voor de inhoud of waarde van een element zijn er regels:

- Elementwaarden mogen niet de ‘geserveerde’ tekens (<, >, &, ' en ") bevatten. Deze geserveerde tekens hebben een speciale functie in XML. Wil je ze toch opnemen, dan moet je het teken vervangen door de volgende code:

Teken	Code
<	<
>	>
&	&
'	'
"	"

- Spaties in XML blijven behouden. Spaties mag je niet in tagnamen gebruiken, maar wel in de overige XML-code.

De volgende tabel laat een paar voorbeelden zien van XML-code.

Code	Opmerking
<genre>Rithm & Blues</genre>	Fout: code bevat het geserveerde teken &
<genre>Rithm & Blues</genre>	Goed: het &-teken is vervangen door &
<genre>Rithm & Blues</genre>	Goed: het element krijgt de waarde 'Rithm & Blues'

8.1.2 XML-schemas

Een XML-schema definieert de bouwblokken van een XML-document. Een welgevormd document moet gevalideerd worden tegen een XML-schema. Een XML-schema kun je vergelijken met het tabelschema van een database: het is een beschrijving van de datastructuur en de data-types van de elementen in een XML-document. Het beschrijft ook de restricties (beperkingen) van je data. In de volgende opgave coderen we een XML-schema-definitie (XSD) die gebaseerd is op het **nieuwetitels.XML** document:

- Opgave** 3 Open je editor en neem de volgende code over. Sla het document op als **nieuwetitels.xsd**.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xss:schema xmlns:xss="http://www.w3.org/2001/XMLSchema">
...
</xss:schema>
```

Naamruimten

De eerste regel uit de vorige opgave is de standaard XML-declaratie. De tweede regel is het schema-element. Het schema-element is het root-element van een XML-schema. In dit element moeten we als volgt een naamruimte (of ‘namespace’) definiëren:

```
xmlns:namespace
```

Bijvoorbeeld:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

In deze regel is de naam van de naamruimte ‘xs’. Met de Uniform Resource Identifier (URI) <http://www.w3.org/2001/XMLSchema> geef je aan dat het document moet gevalideerd worden tegen een XML-schema.

Met naamruimten voorkom je naambotsing of naamconflicten.

Naambotsing gebeurt wanneer er elementen uit verschillende documenten dezelfde naam krijgen. Bijvoorbeeld, in het ene document beschrijft het element `<table>` een tafel (het meubel). In een tweede document beschrijft het element `<table>` een HTML-tag. Als we een element associëren met een naamruimte dan is er geen misverstand en treedt er geen naambotsing op.

Simple en complexe elementen

Een simpel element is een element zonder subelementen. Een complex element is een element met subelementen. In de volgende opgave is het root-element een complex element.

- Opgave** 4 Open **nieuwetitels.xsd** en voeg de definitie van het **nieuwetitels**-element uit je **nieuwetitels.XML** eraan toe:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="nieuwetitels">
    <xs:complexType>
      <xs:sequence>
        ....
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

8.1.3 Datatypes

Ook in XML kennen we datatypes. De meeste voorkomende datatypes in schema’s zijn:

- string
- decimal
- integer

- boolean
- date
- time

In de volgende stap definiëren we het leverancier-element uit het XML-document. Het leverancier-element is van het complexe type, omdat het de subelementen naam, code en datum heeft. De volgorde van de subelementen woord aangegeven binnen de **sequence** tags.

Opgave 5 Open **nieuwetitels.xsd** en voeg de definitie van het leverancier-element eraan toe:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xss:schema xmlns:xss="http://www.w3.org/2001/XMLSchema">
<xss:element name="nieuwetitels">
<xss:complexType>
<xss:sequence>

<xss:element name="leverancier">
<xss:complexType>
<xss:sequence>
<xss:element name="naam" type="xs:string" />
<xss:element name="code" type="xs:string" />
<xss:element name="datum" type="xs:date" />
</xss:sequence>
</xss:complexType>
</xss:leverancier>

</xss:sequence>
</xss:complexType>
</xss:nieuwetitels>
</xss:schema>
```

De subelementen naam, code en datum zijn van het simpele type, want zij hebben geen subelementen.

Het datatype: date

Het datatype date gebruiken we voor datum gegevens, bijvoorbeeld:

```
<xss:element name="datum" type="xs:date" />
```

Het datatype date heeft het volgende formaat: JJJJ-MM-DD, bijvoorbeeld 2010-10-26. We kunnen het datum-element als volgt verder definiëren:

```
<xss:element name="jaar" type="xs:gYear" />
<xss:element name="maand" type="xs:gMonth" />
<xss:element name="dag" type="xs:gDay" />
```

Het datatype: time

Het datatype time gebruiken we voor tijd gegevens en krijgt het volgende formaat:

hh:mm:ss, bijvoorbeeld 15:23:26/

Het datatype: dateTime

Het datatype `dateTime` gebruiken we voor datum en tijd gegevens en krijgt het volgende formaat:

`JJJJ-MM-DDThh:mm:ss`, bijvoorbeeld `2010-11-03T09:23:26`.

Numerieke datatypes

De volgende datatypes definiëren numerieke waarden:

- **byte** is een 8-bit integer waarde (positief of negatief)
- **decimal** is een decimale waarde (positief of negatief)
- **int** is een 32-bit integer waarde (positief of negatief)
- **integer** is een integer waarde (positief of negatief)
- **long** is een 64-bit integer waarde (positief of negatief)
- **negativeInteger** is een integer met alleen negatieve waarden
- **positiveInteger** is een integer met alleen positieve waarden
- **short** is een 16-bit integer waarde (positief of negatief)

Het datatype: boolean

Het datatype `boolean` specificiert een true of false waarde. De mogelijke waarden zijn true en false, of 1 (hetzelfde als true) en 0 (hetzelfde als false).

8.1.4 Indicatoren

Er zijn twee soorten indicatoren om de vorm van het document te controleren:

- Order indicatoren: All, Choice, Sequence
- Occurrence indicatoren: `maxOccurs`, `minOccurs`

In de volgende opgave definiëren we het boek-element. Dit element mag meerdere keer voorkomen binnen het document nieuwetitels. Hiervoor gebruiken we het attribuut `maxOccurs` als ‘unbounded’. `maxOccurs` is het maximale en `minOccurs` is het minimale aantal keer het element mag voorkomen.

Opgave 6 Open **nieuwetitels.xsd** en voeg de definitie van het boekelement eraan toe:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xss:schema xmlns:xss="http://www.w3.org/2001/XMLSchema">
<xss:element name="nieuwetitels">
<xss:complexType>
<xss:sequence>
<xss:element name="leverancier">
<xss:complexType>
<xss:sequence>
<xss:element name="naam" type="xs:string" />
<xss:element name="code" type="xs:string" />
<xss:element name="datum" type="xs:dateTime" />
</xss:sequence>
</xss:complexType>
</xss:leverancier>
```

```
<xs:element name="boek" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="isbn" type="xs:string" />
<xs:element name="uitgever" type="xs:string" />
<xs:element name="auteur" type="xs:string" />
<xs:element name="titel" type="xs:string" />
<xs:element name="genre" type="xs:string" />
<xs:element name="cover" type="xs:string" />
<xs:element name="prijs" type="xs:decimal" />
<xs:element name="korting" type="xs:boolean" />
<xs:element name="aantal" type="xs:positiveInteger" />
</xs:sequence>
</xs:complexType>
</xs:boek>

</xs:sequence>
</xs:complexType>
</xs:nieuwetitels>
</xs:schema>
```

8.1.5 Restricties voor datatypes

We kunnen een aantal restricties of beperkingen voor datatypes definiëren. Met restricties kunnen we ervoor zorgen voor een stricte validatie van de data in het document.

Enumeration (opsomming)

Enumeration is een lijst met acceptabele waarden. In de volgende opgave definiëren we een opsomming van de acceptabele waarden voor het genre-element.

- Opgave** 7 Open **nieuwetitels.xsd** en voeg de definitie van het genre-element eraan toe:

```
<xs:element name="genre">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="ict"/>
<xs:enumeration value="literatuur"/>
<xs:enumeration value="psychologie"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
```

fractionDigits (decimale posities)

fractionDigits definieert het acceptabele aantal decimale posities in een element. De waarde moet nul of groter zijn.

- Opgave 8** Open **nieuwetitels.xsd** en voeg de definitie van het prijs-element eraan toe:

```
<xs:element name="prijs">
    <xs:simpleType>
        <xs:restriction base="xs:decimal">
            <xs:fractionDigits value="2"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Length (lengte)

Length definieert de acceptabele lengte in tekens van een element. De waarde moet nul of groter zijn.

- Opgave 9** Open **nieuwetitels.xsd** en voeg de definitie van het isbn-element eraan toe:

```
<xs:element name="isbn">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:length value="10"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

De volgende restricties zijn van toepassing op de lengte van elementen:

- maxExclusive: de waarde moet minder dan maxExclusive zijn
- minExclusive: de waarde moet groter dan minExclusive zijn;
- maxInclusive: de waarde mag t/m maxInclusive zijn;
- minInclusive: de waarde mag vanaf minInclusive zijn;
- maxLength: de lengte mag t/m maxLength zijn;
- minLength: de lengte mag vanaf minLength zijn.

- Opgave 10** Open **nieuwetitels.xsd** en voeg de definitie van het isbn-element eraan toe:

```
<xs:element name="aantal">
    <xs:simpleType>
        <xs:restriction base="xs:positiveInteger">
            <xs:minInclusive value="50"/>
            <xs:maxInclusive value="100"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Default (standaardinstelling)

Een element krijgt de default-waarde wanneer er geen andere waarde is gespecificeerd.

Opgave 11 Open **nieuwetitels.xsd** en voeg de definitie van het uitgever-element eraan toe:

```
<xs:element name="uitgever" type="xs:string"  
default="Academic Service" />
```

Required (verplicht)

Een element met de required attribuut betekent dat de data verplicht is.

Opgave 12 Open **nieuwetitels.xsd** en voeg de definitie van het isbn-element eraan toe:

```
<xs:element name="isbn" use="required">
```

Paterns (patronen)

Om input data nog strikter te controleren gebruiken we patronen.

Patronen definiëren we als volgt:

- [a-z] betekent een teken van uit de verzameling van kleineletters (a t/m z)
- [A-Z] betekent een teken van uit de verzameling van hoofdletters (A t/m Z)
- [0-9] betekent een getal van uit de verzameling (0 t/m 9)
- {3} betekent een patroon met drie tekens
- * achter een patroon betekent ‘nul of meer keer herhalen.’

Een paar voorbeelden:

[0-9]{2} betekent alle mogelijke combinaties van twee cijfers

<xs:pattern value="([a-zA-Z])*/> valideert nul of meer kleine of hoofdletters

<xs:pattern value="man/vrouw"/> valideert man of vrouw:

In de volgende opgave valideren we input voor het code-element. De input mag vijf cijfers, kleine letters of hoofdletters bevatten.

Opgave 13 Open **nieuwetitels.xsd** en voeg de definitie van het code-element eraan toe:

```
<xs:element name="code">  
  <xs:simpleType>  
    <xs:restriction base="xs:string">  
      <xs:pattern value="[a-zA-Z0-9]{5}"/>  
    </xs:restriction>  
  </xs:simpleType>  
</xs:element>
```

8.1.6 Meer over XML-schema's

XML Schema's zijn uitbreidbaar

XML-schema's zijn uitbreidbaar ('extensible'), dus je kunt later altijd nog informatie toevoegen. Met extensible XML-schema's mag je:

- schema's binnen andere schema's gebruiken;
- data types gebaseerd op andere data types maken;
- meerdere schema's in een document gebruiken.

Link vanuit XML-document naar XML-schema

Om een schema te gebruiken moet je een link maken van een XML-document naar een XML-schema. In de volgende opgave gaan we een link maken vanuit ons brondocument **nieuwetitels.XML** naar ons XML-schema. Dat doen we met het schemaLocation attribuut.

Opgave 14 Open **nieuwetitels.XML** en voeg de link naar **nieuwetitels.xsd** eraan toe:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<nieuwetitels
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xs:schemaLocation="nieuwetitels.xsd">
  .
  .
  .
</nieuwetitels>
```

In bovenstaande code zie je de regel `xmlns:xs="http://www.w3.org/2001/XMLSchema"`. Deze geeft aan dat de elementen en datatypes in dit document gevalideerd moeten worden met een XML-schema. Het geeft ook aan dat de naamruimte 'xs' is. Alle elementen en datatypes moeten deze naamruimte als prefix krijgen.

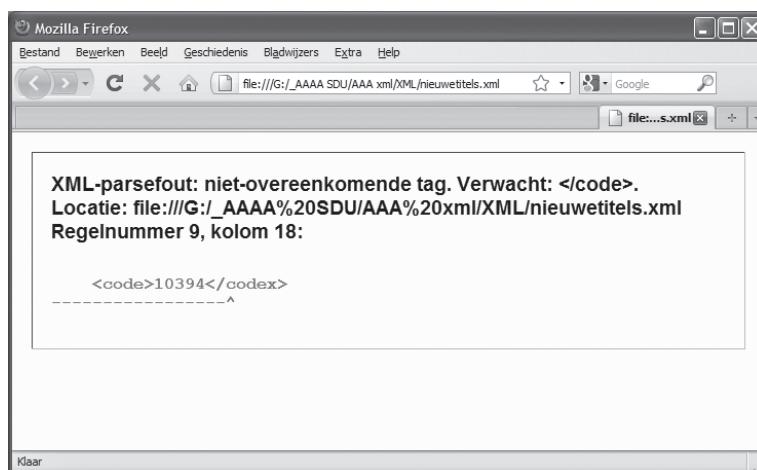
De regel `xs:schemaLocation="nieuwetitels.xsd"` geeft aan dat het schema te vinden is in dezelfde map van het XML-document.

XML valideren

We kunnen nu controleren of het XML-document wel gevormd is. Een wel gevormd XML-document is een document zonder syntaxfouten. Als er fouten in je document zitten, kun je dat zien als je het document met de browser opent. Je krijgt dan een foutmelding te zien, zoals in figuur 8.4.

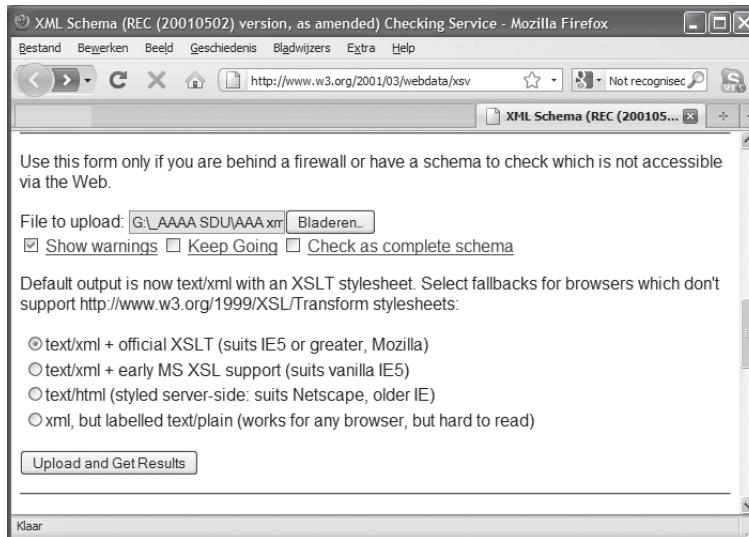
Figuur 8.4

De browser meldt een fout in een XML-document



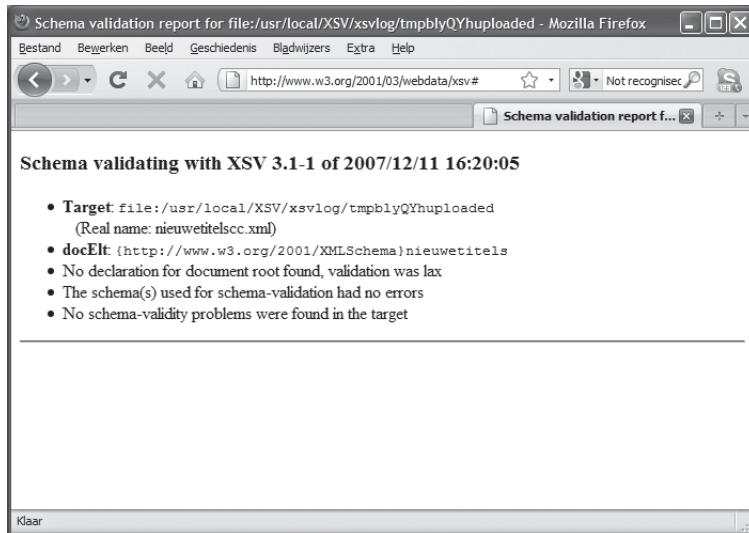
Er is ook een betere manier om vast te stellen of je XML-document wel gevormd is: door het te laten controleren ('valideren') met een speciaal programma (een validator). De W3org stelt een XML-validator beschikbaar via hun website: www.w3.org/2001/03/webdata/xsv. Halverwege deze webpagina heb je de mogelijkheid om een bestand te uploaden (zie figuur 8.5).

Figuur 8.5
De XML-schema validator van w3org



Als je klikt op de knop *Bladeren*, kun je op je eigen computer je de locatie zoeken van je XML-document dat je wilt laten valideren. Vink *Show warnings* aan en klik op *Upload and Get Results*. In figuur 8.6 zie je het resultaat van een goed gevalideerd XML-document.

Figuur 8.6
Het gavalideerde XML-document bevat geen fouten



XML code camp - Lab 1

Nieuwe titels

Open **nieuwetitels.XML** en voeg er negen nieuwe titels aan toe.

8.2 XML en PHP

Als softwareontwikkelaar zul je soms overwegen data te bewaren in XML-formaat in plaats van in een database. Het hangt van de te bouwen applicatie af wat voor datastructuren de beste oplossing kan zijn. PHP en veel andere programmeertalen hebben zich verder ontwikkeld in het ondersteunen van XML. In deze paragraaf kijken we naar een eenvoudige manier om XML-data in PHP te verwerken.

8.2.1 SimpleXML

Om het verwerken van XML-documenten binnen PHP eenvoudiger te maken, heeft Sterling Hughes de uitbreiding SimpleXML in PHP ontwikkeld. Met SimpleXML heb je een aantal functies tot je beschikking die het werken met XML-bestanden mogelijk maken. We gaan hier de volgende functies bekijken:

- `SimpleXML_load_file()`
- `SimpleXML_load_string()`
- `Xpath()`
- `asXML()`

Aan de hand van het bestand **nieuwetitels.XML** gaan we kennismaken met de mogelijkheden van SimpleXML.

Opgave 15 Open het bestand **nieuwetitels.XML** dat je hebt gemaakt in de vorige paragraaf. Als je het bestand niet kunt vinden dan typ je de volgende over en sla het op als **nieuwetitels.XML**.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<nieuwetitels>
    <leverancier>
        <naam>Boeken-Plus</naam>
        <order>AAX394</order>
        <date>2010-10-10</date>
    </leverancier>
    <boek>
        <isbn>9789039525371</isbn>
        <uitgever>Academic Service</uitgever>
        <auteur>G Sanchez Cano</auteur>
        <titel>Maatwerk MBO ICT Webdesign</titel>
        <genre>ict</genre>
        <cover>http://SDU.images/webdesign/large.jpg</cover>
        <prijs>29.95</prijs>
        <korting>true</korting>
        <aantal>100</aantal>
    </boek>
    <boek>
```

```
<isbn>9780321573827</isbn>
<uitgever>Adobe Press</uitgever>
<auteur>Russell Chun</auteur>
<titel>Adobe Flash CD4 Professional</titel>
<genre>webdesign</genre>
<cover>http://adobe.images/webdesign/large.jpg
</cover>
<prijs>59.95</prijs>
<korting>false</korting>
<aantal>130</aantal>
</boek>
<boek>
<isbn>9789039525906</isbn>
<uitgever>Academic Service</uitgever>
<auteur>Hans van Rheenen, G Sanchez Cano, Bert
Pinkster</auteur> <titel>Maatwerk MBO ICT
Basisboek 3 ICT</titel>
<genre>ict</genre>
<cover>http://SDU.images/basisboek3/large.jpg
</cover>
<prijs>39.95</prijs>
<korting>true</korting>
<aantal>200</aantal>
</boek>
</nieuwetitels>
```

Het root-element is `<nieuwetitels>` met nieuwe boektitels. Elke boektitel heeft een isbn, uitgever, auteur, titel, genre, cover, prijs, korting, en aantal.

8.2.2 simpleXML_load_file()

Om je XML-bestanden te kunnen verwerken, moeten we eerst het bestand in PHP ‘laden.’ Dit doen we met de functie `simpleXML_load_file()`. Deze functie gebruikt de volgende syntaxis:

```
simpleXML_load_file('bestandnaam.XML');
```

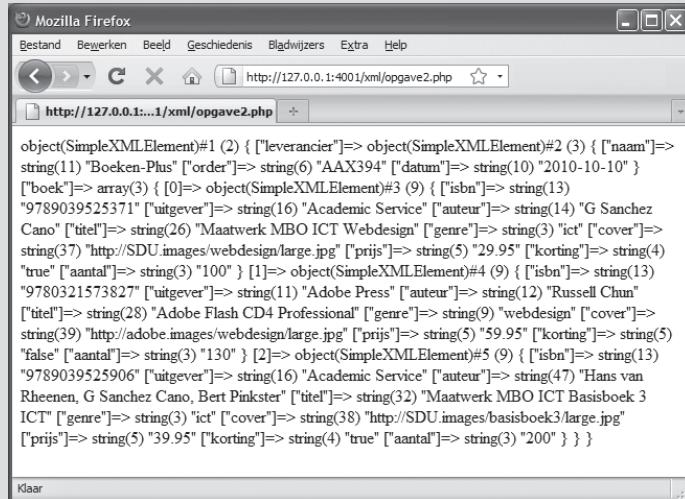
Opgave 16 Open je editor en typ het volgende php-script over. Sla het script op als **nieuwetitels.php**.

```
<?php
    if (file_exists('nieuwetitels.xml'))
    {
        $XML = simpleXML_load_file('nieuwetitels.xml');
        var_dump($XML);
    }
?>
```

Het resultaat moet er zo uitzien:

Figuur 8.7

Resultaat van
var_dump



```

object(SimpleXMLElement)#1 (2) { ["leverancier"]=> object(SimpleXMLElement)#2 (3) { ["naam"]=> string(11) "Boeken-Plus" ["order"]=> string(6) "AAX394" ["datum"]=> string(10) "2010-10-10" } ["boek"]=> array(3) { [0]=> object(SimpleXMLElement)#3 (9) { ["isbn"]=> string(13) "9789039525371" ["uitgever"]=> string(16) "Academic Service" ["auteur"]=> string(14) "G Sanchez Cano" ["titel"]=> string(26) "Maaatwerk MBO ICT Webdesign" ["genre"]=> string(3) "ict" ["cover"]=> string(37) "http://SDU/images/webdesign/large.jpg" ["prijs"]=> string(5) "29.95" ["korting"]=> string(4) "true" ["aantal"]=> string(3) "100" } [1]=> object(SimpleXMLElement)#4 (9) { ["isbn"]=> string(13) "9780321573827" ["uitgever"]=> string(11) "Adobe Press" ["auteur"]=> string(12) "Russell Chun" ["titel"]=> string(28) "Adobe Flash CD4 Professional" ["genre"]=> string(9) "webdesign" ["cover"]=> string(39) "http://adobe.images/webdesign/large.jpg" ["prijs"]=> string(5) "59.95" ["korting"]=> string(5) "false" ["aantal"]=> string(3) "130" } [2]=> object(SimpleXMLElement)#5 (9) { ["isbn"]=> string(13) "9789039525906" ["uitgever"]=> string(16) "Academic Service" ["auteur"]=> string(47) "Hans van Rheenen, G Sanchez Cano, Bert Pinkster" ["titel"]=> string(32) "Maaatwerk MBO ICT Basisboek 3 ICT" ["genre"]=> string(3) "ict" ["cover"]=> string(38) "http://SDU/images/basisboek3/large.jpg" ["prijs"]=> string(5) "39.95" ["korting"]=> string(4) "true" ["aantal"]=> string(3) "200" } }

```

In figuur 8.7 zie je dat het object `simpleXMLElement` vier objecten heeft. Het eerste object is de array `leverancier` met daarin drie sub-elementen, etc.

8.2.3 De foreach-lus

Met de `foreach`-lus kunnen we alle elementen in het bestand lezen.

Opgave 17 Open `nieuwetitels.php` en voeg de onderstaande codes eraan toe.

```

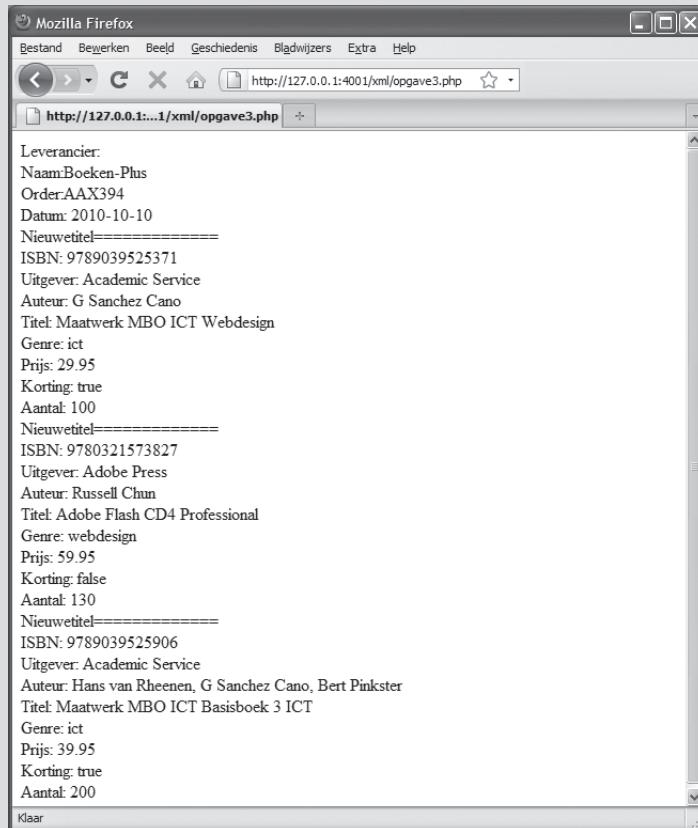
<?php
    $XML = simpleXML _ load _ file('nieuwetitels.xml');
    foreach ($XML->leverancier as $leverancier)
    {
        echo "Leverancier:<br />
Naam:{$leverancier->naam}<br />
Order:{$leverancier->order}<br />
Datum: {$leverancier->datum}<br />";
    }
    foreach ($XML->boek as $boek)
    {
        echo "Nieuwetitel======<br />
ISBN: {$boek->isbn}<br />
Uitgever: {$boek->uitgever}<br />
Auteur: {$boek->auteur}<br />
Titel: {$boek->titel}<br />
Genre: {$boek->genre}<br />
Prijs: {$boek->prijs}<br />
Korting: {$boek->korting}<br />
Aantal: {$boek->aantal}<br />";
    }
?>

```

Het resultaat moet er zo uitzien:

Figuur 8.8

Nieuwe titels
weergeven



De eerste stap in het script van de vorige opgave is het laden van het XML-bestand. In deze stap worden alle XML-elementen opgenomen in php-variabelen:

```
$XML = simpleXML_load_file('nieuwetitels.xml');
```

De volgende stap is het weergeven van het leverancier-element met de foreach-lus:

```
foreach ($XML->leverancier as $leverancier)
```

De laatste stap is het weergeven van alle nieuwetitels, ook weer met de foreach-lus:

```
foreach ($XML->boek as $boek)
```

8.2.4 simpleXML_load_string()

Stel je voor dat we constante gegevens apart willen hebben, bijvoorbeeld in de vorm van een string. In deze gevallen kunnen we gebruik maken van de functie `simpleXML_load_string()`. Hiermee kunnen we gegevens vanuit een string met een XML-formaat laden.

Opgave 18 Typ het volgende php-script over en sla het op als **opgave18.php**.

```
<?php
$XMLstring = <<<XML
<?xml version="1.0" encoding="ISO-8859-1"?>
<root>
<leverancier>
    <code>0001</code>
    <naam>Boeken-Plus</naam>
</leverancier>
<leverancier>
    <code>0002</code>
    <naam>Book World</naam>
</leverancier>
</root>
XML;
$XML = simpleXML_load_string($XMLstring);
var_dump($XML);
foreach($XML->leverancier as $leverancier)
{
    echo "Code: {$leverancier->code} <br />
        Naam: {$leverancier->naam} <br />";
}
?>
```

8.2.5 XML-queries met XPath()

Met de functie `XPath()` kunnen we queries of SELECT-opdrachten uitvoeren op een XML-document. Het resultaat is een array-verzameling met de geselecteerde elementen. De syntaxis is als volgt:

```
array xpath(string path)
```

Bijvoorbeeld, het selecteren van alle auteurs doe je als volgt:

```
$auteurs = $XML->xpath('/nieuwetitels/boek/auteur');
```

Opgave 19 Typ het volgende php-script over en sla het op als **opgave19.php**.

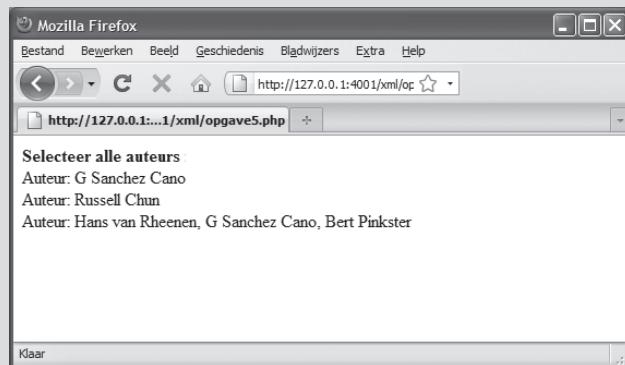
```
<?php
$XML = simpleXML_load_file('nieuwetitels.xml');

echo "<strong>Selecteer alle auteurs</strong>
<br />";
$auteurs = $XML->xpath('/nieuwetitels/boek/auteur');
foreach($auteurs as $auteur)
{
    echo "Auteur: $auteur<br />";
}
?>
```

Het resultaat moet er zo uitzien:

Figuur 8.9

Selecteer alle auteurs



De query of SELECT-opdracht is de input voor de functie `xpath()`. In deze query wordt er alleen gezocht naar de auteur-elementen '`/nieuwetitels/boek/auteur`':

```
$auteurs = $nieuwetitels->xpath('/nieuwetitels/boek/auteur');
```

Het resultaat is de array-verzameling `$auteurs`. Met de foreach-lus kunnen we het resultaat van de query weergeven:

```
foreach($auteurs as $auteur)
{
    echo "Auteur: $auteur<br />";
}
```

In de volgende query selecteren we *alle* boeken. Daar na kunnen we per boek sub-elementen van het boek-element weergeven.

Opgaven 20 Typ het volgende php-script over en sla het op als **opgave20.php**.

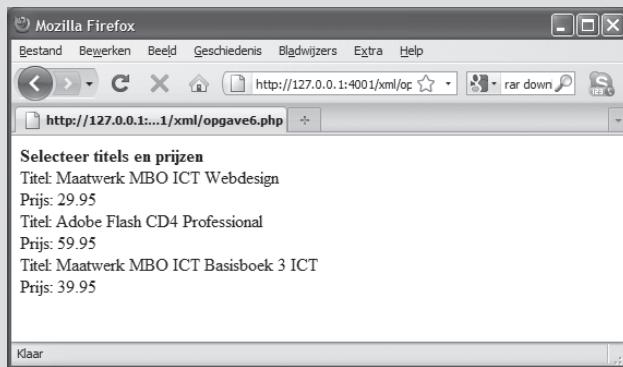
```
<?php
$XML = simpleXML _ load _ file('nieuwetitels.xml');

echo "<strong>Selecteer titels en prijzen
</strong><br />";
$boeken = $XML->xpath('/nieuwetitels/boek');
foreach($boeken as $boek)
{
    echo "Titel: {$boek->titel}<br />
    Prijs: {$boek->prijs}<br />";
}
?>
```

Het resultaat moet er zo uitzien :

Figuur 8.10

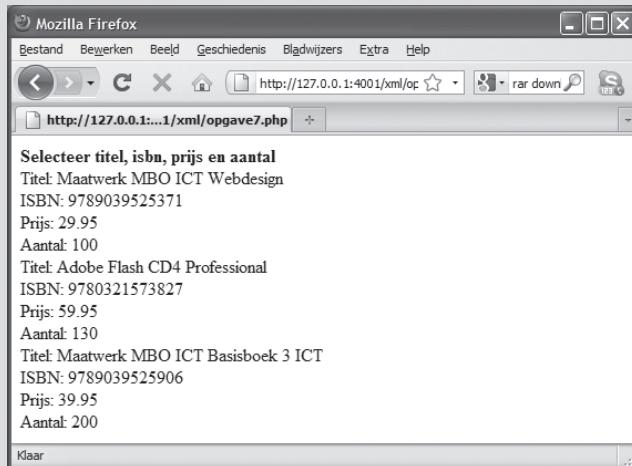
Selecteer titels en prijzen



- 21 Maak een php-script en selecteer titel, isbn, prijs en aantal boeken per titel. Sla het script op als **opgave21.php**. Het resultaat moet er dan zo uitzien:

Figuur 8.11

selecteer titel isbn
prijs en aantal.



Er zijn drie manieren voor het benaderen van een element.

- De eerste manier is om het element rechtstreeks te selecteren door het complete pad te specificeren: `xpath('/nieuwetitels/boek/auteur')`
- De tweede manier is dat je eerst een element rechtstreeks selecteert: `xpath('/nieuwetitels/boek')` en daarna het sub-element als volgt selecteert: `={$boek->auteur}`
- Als laatste kunnen we een globale SELECT-opdracht uitvoeren: `xpath('//auteur')`
Deze opdracht selecteert alle auteur-elementen ongeacht op welk niveau het element gevonden was. Een globale SELECT-opdracht begint met `//`.

8.2.6 Where-clausule met XPath()

Met de WHERE-clausule van de functie `Xpath()` kunnen we, net zoals in SQL, ook een voorwaarde in de SELECT-opdracht specificeren. Syntaxis:

```
xpath('//element[voorwaarde]');
```

Bijvoorbeeld:

```
$boeken = $XML->xpath('//boek[auteur="G Sanchez Cano"]);
```

- Opgaven 22** Typ het volgende php-script over en sla het op als **opgave22.php**.

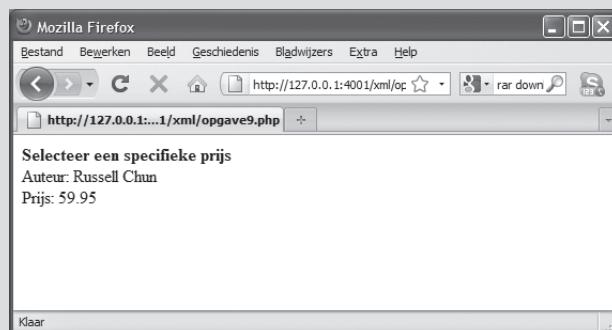
```
<?php
    $XML = simpleXML_load_file('nieuwetitels.xml');

    echo "<strong>Selecteer een specifieke auteur
</strong><br />";
    $boeken = $XML->xpath('//boek[auteur="G Sanchez Cano"]);
    foreach($boeken as $boek)
    {
        echo "Auteur: {$boek->auteur}<br />
        Prijs: {$boek->prijs}<br />";
    }
?>
```

- 23** Maak een php-script en selecteer alle nieuwe titels waar de prijs groter dan 50 euro is. Sla je script op als **opgave23.php**. Het resultaat moet er dan zo uitzien:

Figuur 8.12

Selecteer een specifieke prijs



Het is ook mogelijk om ‘multi-queries’ uit te voeren, bijvoorbeeld.

```
$boeken = $nieuwetitels->xpath('//boek[prijs <=50]
[genre="ict"]);
```

- 24** Typ het volgende php-script over en sla het op als **opgave24.php**.

```
<?php
    $XML = simpleXML_load_file('nieuwetitels.xml');

    echo "<strong>Selecteer een specifieke auteur en genre</
strong><br />";
```

```
$boeken=$XML->xpath('//boek[auteur="G Sanchez Cano"]'
[genre="ict"]');
foreach($boeken as $boek)
{
    echo "Auteur: {$boek->auteur}<br />
Genre: {$boek->genre}<br />";
}
?>
```

- 25** Maak een PHP-script en selecteer alle nieuwe titels waar de prijs lager is dan 50 euro en het genre ‘ict’ is. Sla je script op als **opgave25.php**. Het resultaat moet er dan zo uitzien:

Figuur 8.13

Selecteer
specifieke prijs
en genre



Als je meerdere elementen uit verschillende niveaus wilt selecteren, dan mag je het pipe-symbool | gebruiken, bijvoorbeeld:

```
$resultaten=$nieuwetitels->xpath('//leverancier/naam | //
titel');
```

- 26** Typ het volgende php-script over en sla het op als **opgave26.php**.

```
<?php
$nieuwetitels = simpleXML_load_file('nieuwetitels.xml');

echo "<strong>Selecteer verschillende elementen
</strong><br />";
$resultaten=$nieuwetitels->xpath('//leverancier/naam | //
titel');
foreach($resultaten as $resultaat)
{
    echo $resultaat;
}
?>
```

8.2.7 Update-opdracht met asXML()

Met de functie `asXML()` kunnen we de PHP-variabelen uit de functie `simpleXML_load_file()` terugzetten naar een XML-bestand. In de volgende opgave is het boek *Webdesign* afgeprijsd naar 45 euro. De nieuwe waarden worden met de functie `asXML()` teruggeschreven naar het XML-bestand. Bijvoorbeeld:

```
$XML->asXML('nieuwetitels.xml');
```

Opgave 27 Typ het volgende php-script over en sla het op als **opgave27.php**.

```
<?php
$XML = simpleXML_load_file('nieuwetitels.XML');
echo "<strong>Update de prijs van een bepaald
boek</strong><br />";
$resultaat=$XML->xpath(
'//boek[isbn="9780321573827"]');
$resultaat[0]->prijs = 45;
$XML->asXML('nieuwetitels.XML');
?>
```

8.2.8 Transformeren van XML met XSLT

XSL staat voor EXtensible Stylesheet Language. Dit is de style-sheet scriptingtaal voor XML- documenten. De T in XSLT staat voor Transformeren. XSLT is een scriptingtaal die gebaseerd is op XML. Met XSLT kunnen we een XML-bestand transformeren naar een XHTML, WAP of SQL output-formaat dat is gebaseerd op de gegeven styles in het XSL-bestand.

Opgave 28 Typ het volgende script over en sla het op als **nieuwetitels.xsl**.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
<html xmlns="http://www.w3.org/1999/xhtml" lang="nl">
<head>
<title>XSLT</title>
</head>
<body>
<h2>Nieuwetitels</h2>
<xsl:for-each select="nieuwetitels/leverancier">
<h3><xsl:value-of select="naam"/></h3>
</xsl:for-each>

<table border="1">
<tr bgcolor="#9acd32">
<th>isbn</th>
<th>uitgever</th>
<th>auteur</th>
<th>titel</th>
<th>genre</th>
<th>prijs</th>
<th>korting</th>
<th>aantal</th>
</tr>
<xsl:for-each select="nieuwetitels/boek">
<tr>
<td><xsl:value-of select="isbn"/></td>
```

```

<td><xsl:value-of select="uitgever"/></td>
<td><xsl:value-of select="auteur"/></td>
<td><xsl:value-of select="titel"/></td>
<td><xsl:value-of select="genre"/></td>
<td><xsl:value-of select="prijs"/></td>
<td><xsl:value-of select="korting"/></td>
<td><xsl:value-of select="aantal"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

8.2.9 Link van XML naar XSLT

Om een link te maken van een XML-bestand naar een XSLT-bestand is er maar een regel code nodig. In de volgende opgave gaan we dat doen.

Opgave 29 Open **nieuwetitels.xml** en voeg de tweede regel code eraan toe.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<?xml-stylesheet type="text/xsl" href="nieuwetitels.xsl"?>
```

Figuur 8.14

Resultaat van het transformeren van nieuwe titels naar XHTML

The screenshot shows a Mozilla Firefox window with the title 'XSLT - Mozilla Firefox'. The address bar displays 'http://127.0.0.1:4001/xml/nieuwetitels.xml'. The page content is titled 'Nieuwetitels' and contains a table with the heading 'Boeken-Plus'. The table has eight columns: isbn, uitgever, auteur, titel, genre, prijs, korting, and aantal. The data is as follows:

isbn	uitgever	auteur	titel	genre	prijs	korting	aantal
9789039525371	Academic Service	G Sanchez Cano	Maatwerk MBO ICT Webdesign	ict	29.95	true	100
9780321573827	Adobe Press	Russell Chun	Adobe Flash CD4 Professional	webdesign	59.95	false	130
9789039525906	Academic Service	Hans van Rheenen, G Sanchez Cano, Bert Pinkster	Maatwerk MBO ICT Basisboek 3 ICT	ict	39.95	true	200

At the bottom of the page, there is a status bar with the word 'Klaar'.

8.2.10 XML en MySQL

Een van onze leveranciers heeft ons een XML-bestand gestuurd. Dit bestand is een order met daarin een aantal nieuwe boektitels. In deze paragraaf willen we alle nieuwe boektitels van onze leverancier Boeken-plus vanuit het bestand **nieuwetitels.XML** importeren onze MySQL database importeren. In deze eerste stap maken we onze database en tabel aan.

Opgave 30 Typ het volgende php-script over en sla het op als **opgave30.php**.

```
<?php
// Hier maak je verbinding met de dataserver.
$verbinding = mysql_connect("localhost","root","");
or die ("Kon geen verbinding maken");

// Hier creëer je de libris database
$query = "CREATE DATABASE libris";

$resultaat= mysql_query($query, $verbinding) or
die ( mysql_error() );

// Hier selecteer je de libris database
mysql_select_db('libris')
or die("Kon geen database selecteren");

// Hier maak je de boekentabel aan.
$query = "CREATE TABLE boeken (
    isbn varchar(13),
    uitgever varchar(30),
    auteur varchar(50),
    titel varchar(50),
    genre varchar(20),
    cover varchar(50),
    prijs decimal(10,2)NOT NULL default '0.00',
    voorraad integer(3) NOT NULL default '0',
    PRIMARY KEY  (isbn)
)";
$resultaat= mysql_query($query, $verbinding) or
die ( mysql_error() );
echo"Libris database en boeken tabel aangemaakt.";
?>
```

In de volgende stap maken we een php-script dat het XML-bestand leest en de data naar de libris-database importeert.

Opgave 31 Typ het volgende php-script over en sla het op als **opgave31.php**.

```
<?php
$XML_bestand = 'nieuwetitels.xml';
if (file_exists($XML_bestand))
{
    $XML = simpleXML_load_file($XML_bestand);
}
else
{
    exit('Error: kon bestand '.$XML.' niet openen.');
}

$verbinding = mysql_connect('localhost', 'root', '')
or die ('Error: kon geen verbinding maken');
```

```

mysql _ select _ db('libris')
or die('Error: kon geen database selecteren');

// Hier lezen we nieuwetitels.xml
$teller=0;
foreach ($XML->boek as $boek)
{
    // Hier importeren we alle nieuwetitels
    $query = "INSERT INTO boeken "
        "(isbn,uitgever,auteur,titel,genre,cover,prijs,voorra
        ad)";
    "$VALUES ('$boek->isbn', ".
    "'$boek->uitgever', ".
    "'$boek->auteur', ".
    "'$boek->titel', ".
    "'$boek->genre', ".
    "'$boek->cover', ".
    "'$boek->prijs', ".
    "'$boek->aantal')";
    $resultaat= mysql _ query($query, $verbinding) or die
        ( mysql _ error( ) );
    // Nieuwetitels optellen
    $teller++;
}
echo "<br/>";
echo "-----<br/>";
echo "Aantal geimporteerde nieuwe titels: $teller<br/>";
echo "-----<br/>";
?>

```

isbn	uitgever	auteur	titel	genre	cover	prijs	voorraad
9789039525371	Academic Service	G Sanchez Cano	Maatwerk MBO ICT Webdesign	ict	http://SDU.images/webdesign/large.jpg	29.95	100
9780321573827	Adobe Press	Russell Chun	Adobe Flash CD4 Professional	webdesign	http://adobe.images/webdesign/large.jpg	59.95	130
9789039525906	Academic Service	Hans van Rheenen, G Sanchez Cano, Bert Pinkster	Maatwerk MBO ICT Basisboek 3	ict	http://SDU.images/basisboek3/large.jpg	39.95	200

Figuur 8.15 Resultaat van het importeren van nieuwetitels.

XML code camp - Lab 2

Nieuwe employees

Maak een XML-bestand en een schema, gebaseerd op de volgende gegevens:

Gegevens voor nieuwe employees:

- employeenummer
- voornaam
- achternaam
- geboortedatum
- begindatum
- adres

Geldige functies zijn:

- junior programmeur
- senior programmeur
- junior software engineer
- senior software engineer
- senior developer
- software architect

- postcode
- plaats
- telefoonnummer
- emailadres
- functie
- salaris

Voeg nu vijf nieuwe employees toe aan het XML-bestand.

XML code camp - Lab 3

In deze toets laat je zien dat je de technieken van PHP, MySQL en XML kunt combineren.

Maak eerst een employee-database in MySQL. Daarna schrijf je een PHP-script voor het importeren van alle nieuwe employees uit het XML-bestand uit XML Lab 2 naar je MySQL employee-database.

Register

- Symbolen**
- \$_POST 190
 - <<extend>> 309
 - <form> 188
 - <<generalization>> 309
 - <<include>> 308
 - <script source> 111
- A**
- aanvraag 171
 - activiteitendiagram 311
 - ADD COLUMN 273
 - ADD PRIMARY KEY 274
 - algoritmes 144
 - ALTER TABLE 273
 - AND 100
 - array 76, 182
 - arguments[] 118
 - declareren 182
 - element verwijderen 86, 87
 - hash 183, 212
 - kopiëren 88
 - multidiimensioneel 78
 - transporteren 88
 - array.join(delimeter) 88
 - array.length 91
 - array-methodes 80
 - array-pointers 81
 - array.pop() 83
 - array-properties
 - testen op 91
 - array.push() 83
 - array.reverse() 85
 - array.shift() 82
 - array.sort() 85
 - array.splice() 84
 - array.unshift() 82
 - asXML() 367
 - attributen 348
 - attribuut 240
- B**
- begintag 349
 - beperkingen voor datatypes 354
- C**
- beslissingsstructuren 97
 - bestand
 - afsluiten 234
 - lezen 235
 - openen 233
 - schrijven 234
 - BETWEEN ... AND 264
 - binary-search 144
 - boolean 72, 180, 353
 - boomstructuur 348
- D**
- CASE 289
 - CHANGE COLUMN 274
 - charAt() 125, 141
 - CHAR_LENGTH() 277
 - clearTimeout() 151
 - clientprogramma 171
 - CONCAT() 276
 - CONCAT_WS() 276
 - confirm() 67
 - constante 217
 - controlestructuur 203
 - cookies 160, 221
 - browser 164
 - JavaScript 161
 - COUNT 268
 - CREATE DATABASE 270
 - CREATE FUNCTION 292
 - CREATE PROCEDURE 285
 - CREATE TABLE 271
 - CREATE TRIGGER 293
 - CREATE VIEW 293
- E**
- database 239
 - maken 256
 - databaseadministrator 260
 - databasemanagementsysteem 255
 - data container 175
 - data modellering 250
 - software 253
 - datastructuur 182, 346
 - datatype
 - boolean 72
- F**
- floating-point 73
 - integer 72
 - NaN 75
 - number 72
 - string 71
 - typeof 72
 - undefined 74
- G**
- datatypes 71, 351
 - beperkingen 354
 - numerieke 353
 - datatypes in PHP 177
 - date 352
 - date() 219
 - date-methodes 96
 - date-object
 - creëren met parameters 94
 - declareren 93
 - date-object creëren met
 - getallen 95
 - date-object creëren met string
 - 94
 - dateTime 353
 - DBA 260
 - DBMS 255
 - DECLARE CURSOR 290
 - DECLARE HANDLER 290
 - declareren
 - array 77
 - declareren van SQL-variabelen 285
 - delete 86
 - DELETE 270
 - erde normaalvorm 242
 - DISTINCT 265
 - Document Object Model 148
 - document.write() 67
 - DOM 148
 - DOM-events 149
 - do-while-lus 208
 - do-while()-lus 134
 - DROP COLUMN 274
 - DROP DATABASE 271
 - DROP PRIMARY KEY 275
 - DROP PROCEDURE 287
 - DROP TABLE 271

E

echo 174
 één-op-éénrelatie 250
 één-op-veelrelatie 251
 eerste normaalvorm 242
 eindtag 349
 eisenanalyse 307
 element 346
 simpel of complex 351
 else 99
 else-if 102
 elseif 186
 embedden 171
 entiteit 240
 Entiteit Relatie Diagram 250
 entiteittype 239
 Enumeration 354
 ERD 250
 eregi() 225
 explode() 230
 expressie
 reguliere 223
 Extensible Markup Language 345
 extensible XML 348
 externe functies 111

F

fclose() 234
 FETCH 290
 fgetcsv() 236
 fgets() 235
FIND_IN_SET() 277
 float 179
 floating-point 73
 floatval() 180
 fopen() 233
 foreach-lus 210, 361
 foreign key 245
 for(in)-lus 127
 for-lus 203
 for()-lus 122
FORMAT() 277
 form-events 153
 formulier
 variabelen 188
 foutconsole in JavaScript 65

fractionDigits 354
 functie 196
 aanroepen 110
 confirm() 67
 date() 219
 document.write() 67
 eregi() 225
 explode() 230
 externe 111
 externe ~ 200
 floatval() 180
 input 109
 intval() 179
 met input/output 109
 met input-parameter,
 aanroepen 121
 mktime() 220
 nl2br() 229
 parseFloat() 73
 parseInt() 73
 printf() 179
 prompt() 67
 strlen() 230
 strpos() 231
 str_replace() 231
 strtolower() 229
 strtoupper() 229
 substr() 230
 time() 220
 trim() 228
 ucfirst() 229
 functie charAt() 125
 functies
 dynamisch uitvoeren 115
 met meerdere parameters
 114
 zelf coderen 108
 fwrite() 234

G

gebeurtenis 149
 gegevens
 constante 242
 elementaire 242
 invoeren 258
 opslaan 232
 opvragen 232
 proces 242

verwerken 189
 zoekopdracht 295
 gegevensanalyse 239
 gegevensbestand 232
 gereserveerde tekens 350
 getElementById 149
 globale variabele 216
 globale variabelen 117

H

hash 183
 hash-array 212
 hash-arrays 89
 hoofdmenu 257

I

if 97, 185
 IF 288
 if-elseif 186
 include() 200
 indexOf() 138
 indicatoren 353
 INNER JOIN 266
 INNER JOIN SELECT 267
 INOUT-parameter 285
 IN-parameter 285
 INSERT() 277
 INSERT INTO 270, 272
 INSTR() 278
 integer 72, 178
 intval() 179

J

JavaScript 63
 cookies 161
 foutcontrole 65
 winkelwagentje 165
 JavaScript-variabelen 68

K

keyboard-events 159
 kolom
 benoemen 257
 kopiëren
 array 88

L

LEFT() 278
LEFT JOIN 266
 length 140

LIMIT 264
LOCATE() 278
lokale variabelen 117
LOWER() 279
LPAD() 279
LTRIM() 279
lus
 do-while() 134
 for() 122
 for(in) 127
 while() 130
lus onderbreken 209
lussen 122

M
Math.abs(x) 135
Math.ceil(x) 135
Math.floor(x) 135
Math.max() 136
Math.min() 136
Math-object 135
Math.random() 137
Math.round() 137
MAX 269
maxOccurs 353
methode
 array 80
MIN 268
mktime() 220
model
 eisen 307
 ontwerpen 307
MODIFY COLUMN 273
mouse-events 153
multidimensionale array 78
MySQL 255
 programmering 294
My_SQL-dataserver
 verbinding maken 294
MySQL en XML 369
MySQL stringfuncties 275

N
naambotsing 351
naamruimte 351
namespace 351
NaN 75
nl2br() 229

normaalvorm
 erde 242, 246
 eerste 242, 244
 nulde 242, 243
 tweede 242, 245
normalisatieproces
 vormen 242
normaliseren 241
 soorten gegevens 242
nulde normaalvorm 242
nul-optimaliteit 252
number 72

O
occurrence indicatoren 353
onderbreken van lus 209
open source 171
operator
 ternary 187
operatoren van variabelen 69
optionaliteit
 nul 252
 volledige 252
OR 100
ORDER BY 262, 297
order indicatoren 353
OUT-parameter 285

P
parameters in procedure 285
parseFloat() 73
parseInt() 73
patronen 356
Personal Home Page 171
PHP 171
 array 182
 code embedden 171
 programmeertaal 173
 programmering 294
 script 172
 syntaxis 173
PHP en XML 359
phpMyAdmin 255, 260
pointers 290
 array 81
POSITION() 279
printf() 179
prompt() 67
pseudocode 145

R
RDBMS 255
reguliere expressie 223
relatie
 één-op-één 250
 één-op-veel 251
 <<extend>> 309
 <<generalization>> 309
 <<include>> 308
 veel-op-veel 251
RENAME 273
REPEAT() 279
REPLACE() 280
request 171
require() 200
restricties voor datatypes 354
resultaatverzameling
 gesorteerd 297

REVERSE() 280
RIGHT() 280
root-element 346
RPAD() 280
RTRIM() 281

S
script 63
SELECT 296
SELECT * 261
SELECT FROM 295
selectie criterium 298
sequencediagram 312
serverprogramma 171
session 236
setTimeout() 151
SHOW 287
SimpleXML 359
simpleXML_load_file() 360
simpleXML_load_string()
 362

sleutel 241
software analyse
 met freeware software 311
software ontwikkeling
 planning 306
SPACE() 281
splice() 87
split() 141
SQL 255

SQL SELECT 295
 SQL-variabelen declareren 285
 statische variabele 217
 stored programma 283
 string 71, 177
 stringfuncties 275
 String functies 228
 string-methodes 138
 strlen() 230
 stroomdiagrammen 97
 strpos() 231
 str_replace() 231
 strtolower() 229
 strtoupper() 229
 Structured Query Language 255
 subexpressie 224
 substr() 142, 230
 substring() 142
 SUBSTRING(FROM) 281
 SUBSTRING(FROM FOR) 281
 SUBSTRING_INDEX() 282
 SUM 269
 superglobale variabele 215
 switch 106, 193
 syntaxis
 PHP 173
 systeem
 modelleren met UML 307

T

taalcomponent 173
 tabel
 kolommen 257
 maken 256
 velden 257
 zoekopdracht gegevens 295
 tag
 <form> 188
 ternary-operator 187
 ternary-operator (?:) 104
 testen op array-properties 91
 time 352
 time() 220

toFixed(n) 74
 toLowerCase() 143
 top-downmodel 239
 toString() 74
 toUpperCase() 143
 transformeren van XML 368
 transporter
 array 88
 tree 348
 trigger 293
 trim() 228
 TRIM() 282
 TRIM.LEADING FROM) 282
 TRIM.TRAILING FROM) 283
 tweede normaalvorm 242
 typeof 72, 87

U

ucfirst() 229
 UML 307
 modelleren systemen 307
 undefined 74
 UPDATE 269
 update-opdracht met asXML() 367
 UPPER() 283
 use case
 notatie 307
 voorbeelden 310
 use case-diagram 307

V

variabele
 \$_POST 190
 globale 216
 namen 175
 PHP 175
 statische 217
 superglobale 215
 uit formulier 188

variabelen
 globale / lokale 117
 JavaScript 68
 namen 68
 operatoren 69

veel-op-veelrelatie 251
 veld
 benoemen 257
 verwijzende sleutel 245
 volledige optionaliteit 252

W

webapplicatieontwikkeling 171
 wel gevormd 349
 WHERE 263, 298
 where-clausule met XPath() 366
 WHERE LIKE 265
 WHILE 289
 while-lus 206
 while()-lus 130
 Window-events 150
 winkelwagentje in JavaScript 165
 workflow 311

X

XML 345
 XML-documenten 346
 XML-editor 347
 XML en MySQL 369
 XML en PHP 359
 XML-queries 363
 XML-schema 350
 XML toepassingen 346
 XML transformeren 368
 XML tree 348
 XML valideren 357
 XPath() 363
 XPath() met where-clausule 366
 XSLT 368

Z

zoekopdracht
 specifieke kolommen 296