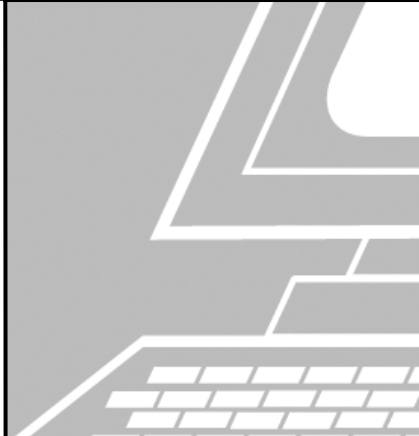




Web



Essentials

Auteur: Johan Cleuren

Lectoren: Johan Cleuren
Tim Dupont
Cedriek Vos
Reinaut Krekels
Carine Derkoningen
Nele Custers

Woord vooraf

Opleidingsonderdeel: Web Essentials			
Opleiding:	Prof. Bachelor Toegepaste informatica		
Onderwijstaal:	Nederlands	Kennisoverdracht:	14 u
Studiepunten:	6	Begeleide kennisverwerking:	42 u
Niveau:	Inleidend	Zelfstudie:	112 u
Lesgevers	Johan Cleuren, Nele Custers, Carine Derkoningen, Tim Dupont, Reinaut Krekels	Studiebelastinguren:	168u

InhoudC

Je leert te werken met de gangbare webtechnologieën. Als inleiding wordt kort de geschiedenis en opbouw van de HTML5-webstandaard behandeld, waarna de lay-out van een webpagina behandeld wordt. Hiervoor worden diverse methoden aangeleerd om met CSS-stijlbladen te werken. Die worden later toegepast op teksten, koppelingen, lijsten, afbeeldingen en tabellen. De CSS-technieken om een pagina in te delen worden uitvoerig behandeld. Ook het multimediale aspect van webpagina's wordt niet vergeten.

Je krijgt een ruime introductie tot standaarden om dynamische webpagina's te creëren, waaronder JavaScript, DOM met W3C Event Handling en PHP.

De einddoelstelling van de cursus is het kunnen coderen van browseronafhankelijke en gevalideerde webpagina's die gebruik maken van huidige webstandaarden. Om je de nodige inzichten te geven in de manier waarop dit alles werkt, zal je rechtstreeks met de code werken door middel van een teksteditor en de browser developer tools.

Leermaterialen

- Deze cursus: Cleuren Johan, Web Essentials, 2017
- <http://bb.pxl.be>: de course Web Essentials dient als ondersteuning van dit opleidingsonderdeel en informatie, updates, aanvullingen, getoonde presentaties, animaties en oefeningen.
- <http://www.pluralsight.com>, Pluralsight video tutorials
- <http://www.codeschool.com>, Code School tutorials en oefeningen
- De contacturen practicum, hou nota's bij.

Achtergrondinformatie

- Smashing HTML5, Bill Sanders, ISBN 978-0-470-97727-9, Wiley, 2011
- <http://www.w3c.com>, referentiesite voor webstandaarden, waaronder CSS.
- <https://whatwg.org>, referentiesite voor HTML 5 en DOM.
- <http://www.php.net/manual>, referentiesite voor PHP.

Evaluatiecriteria/leerdoelen

- De student kan een webpagina opbouwen in HyperText Markup Language (HTML5), bestaande uit semantisch correcte elementen.
- De student kan een webpagina opmaken met Cascading Style Sheets (CSS).

- De student kan een webpagina dynamisch maken met JavaScript en het Document Object Model (DOM).
- De student kan een opdracht zelfstandig opdelen in structuur, opmaak, client-side en server-side taken.
- De student kan een webpagina genereren met PHP: Hypertext Preprocessor (PHP).
- De student kan een HTTP GET- en POST-verzoek verwerken met PHP.
- De student kan scripttaalcode modulair opbouwen met functies.
- De student kan de developer tools van browsers gebruiken bij het ontwikkelen van een website.
- De student kan een webpagina volledig coderen volgens de specificaties van de opdrachtgever.
- De student kan zelfstandig documentatie vinden en verwerken om nieuwe technieken toe te passen in een webpagina.

Evaluatievormen

- Eerste examenkans:
 - Praktijk: Schriftelijk open boek examen met laptop (100%)
- Tweede examenkans:
 - Praktijk: Schriftelijk open boek examen met laptop (100%)

Werkvormen

- Opdrachtgestuurd onderwijs: Aan de hand van duidelijk afgebakende opdrachten/taken verwerven en verwerken studenten de leerinhouden. Het gaat hier niet noodzakelijk om realistische of praktijkopdrachten.

Inhoudsopgave

1	Inleiding HTML5.....	8
1.1	Markeertalen en webstandaarden.....	8
❖	HTML5.....	8
1.2	Basis van een webpagina.....	11
❖	Structuur.....	11
❖	Verklaring van de opbouw.....	12
1.3	Elementen, tags en attributen.....	13
❖	Elementen en hun tags.....	13
❖	Attributen.....	13
❖	Commentaar.....	14
1.4	Verschil tussen de verschillende HTML5-syntaxen.....	14
1.5	Validatie van de code.....	16
1.6	Browsers.....	17
❖	De vier browserengines.....	18
❖	HTML5-ondersteuning.....	18
2	Structuur.....	20
2.1	Blokelementen.....	20
❖	Paragrafen.....	20
❖	Blockquote.....	20
❖	Preformatted.....	21
❖	Koppen.....	21
❖	Adres.....	21
❖	Divisies.....	22
2.2	De structuurelementen van HTML5.....	22
❖	Main.....	23
❖	Secties.....	23
❖	Artikel.....	23
❖	Hoofding.....	23
❖	Voettekst.....	24
❖	Navigatieblok.....	24
❖	Randinformatie.....	24
2.3	Inline-elementen.....	26
❖	Line break.....	26
❖	Word break.....	26
❖	Teksttags.....	26
❖	Logische elementen.....	27
❖	Span.....	28
2.4	Speciale tekens.....	28
3	Hyperlinks.....	29
3.1	Verbindingen leggen.....	29
❖	Opmaak van links.....	29
3.2	Absolute en relatieve koppelingen.....	29
3.3	Interne koppelingen.....	31
3.4	Andere soorten koppelingen.....	32
❖	Koppeling naar een e-mailadres.....	32
❖	Bestanden downloaden.....	32
4	Cascading Style Sheets.....	33
4.1	De syntax van CSS.....	33
❖	Verkorte schrijfwijze CSS.....	33
❖	Commentaar.....	34
❖	Important.....	34

4.2	Toepassing van Cascading Style Sheets.....	34
❖	Lokale stijl: binnen elk HTML-element	34
❖	Globale stijl: in de HTML-pagina.....	34
❖	Gelinkte stijl: apart stijlblad	35
4.3	De waterval en overerving	35
4.4	Classes.....	36
❖	Klassen (classes).....	36
❖	Afhankelijke klassen (fixed classes)	37
❖	Pseudoklassen	38
❖	Pseudoklassen II: Fixed Pseudo-classes	39
❖	Pseudo-elementen	39
4.5	ID's	39
4.6	Speciale selectors	40
❖	Asterisk	40
❖	Gecombineerde klassen.....	40
❖	Contextuele selectors.....	40
❖	Kinderen combineren	41
❖	Attribuutselectors.....	41
4.7	Eenheden	41
4.8	Mediatypes.....	42
5	Opmaak	43
5.1	Kleuren	43
❖	Kleurnamen.....	43
❖	RGB-model	43
❖	HSL-model.....	44
❖	Webveilig kleurenpalet.....	44
5.2	Tekstopmaak	44
❖	Het lettertype.....	44
❖	Opmaak van de tekst.....	46
❖	Invoegen van een lettertype	46
5.3	Het Box-model	46
5.4	Lijnen	49
5.5	Afwijken van het standaard gedrag met CSS.....	50
6	Lijsten.....	51
6.1	Soorten lijsten.....	51
❖	Genummerde lijsten (geordende lijsten)	51
❖	Ongenummerde lijsten	52
❖	Definitielijsten.....	53
6.2	Geneste lijsten	53
6.3	CSS voor lijsten	54
7	Afbeeldingen.....	56
7.1	Bestandsformaten voor het web	56
7.2	Het invoegen van afbeeldingen	56
❖	Het -element	56
❖	Het <figure>-element.....	57
❖	Het <object>-element	57
7.3	De positie van de afbeelding.....	58
❖	Afbeeldingen en tekst	58
❖	Afbeelding als achtergrond	59
❖	Afbeelding als koppeling.....	60
❖	Image maps	60
7.4	Afbeelding klaarmaken voor het web.....	61

❖ Bestandsgrootte	62
8 Audio en video	63
8.1 Geluid.....	63
❖ Geluidsbestanden.....	63
❖ Geluid invoegen: audio.....	63
❖ Alternatieven: source.....	63
8.2 Film.....	64
❖ Videobestanden.....	64
❖ Video invoegen: video	64
8.3 Andere interactieve inhoud	65
❖ Invoegen van animaties: embed	65
❖ Invoegen van andere inhoud: object.....	65
❖ Invoegen van Youtube-filmpjes	67
9 Tabellen	68
9.1 Structuur van een tabel	68
❖ De tabel.....	68
❖ De tabelrij.....	68
❖ De cellen	68
❖ Het bijschrift	69
9.2 Attributen	69
❖ Attributen voor tabellen en rijen	69
❖ Attributen voor cellen.....	70
❖ CSS voor tabellen en cellen.....	70
9.3 Rijgroepen en kolomgroepen	72
❖ Rijgroepen	72
❖ Kolomgroepen.....	73
9.4 Overspannen van kolommen en rijen	74
9.5 Opeenvolgende tabellen.....	75
9.6 Geneste tabellen.....	75
9.7 CSS3 voor tabellen.....	76
10 Positioneren met CSS	78
10.1 Waarom CSS positioning	78
10.2 Blok en inline-elementen.....	79
10.3 Drijvende opmaak	79
10.4 Absolute opmaak	79
10.5 Relatieve opmaak	83
10.6 Vaste positie.....	84
10.7 Stapelvolgorde	85
11 Formulieren	86
11.1 Omschrijving	86
11.2 Elementen en attributen	87
❖ Het <form>-element.....	87
11.3 Invoerelementen	88
❖ Tekstvelden: text	88
❖ Wachtwoordvelden	89
❖ Hidden velden.....	89
❖ E-mailvelden.....	89
❖ Nummervelden	89
❖ Tekstinvoer: url	90
❖ Datumveld: date, month, week, time, datetime, datetime-local	90
❖ Zoekveld: search	90
❖ Range velden	91
❖ File velden	91

❖	Selectievakken: checkbox	91
❖	Keuzerondjes: radiobuttons	91
❖	Kleurenveld: color	92
❖	Uitvoerveld: output	92
11.4	Keuzelijsten	92
❖	Dropdownmenu: select	92
11.5	Tekstvakken	93
❖	Tekstveld met meerdere regels: textarea	93
11.6	Knoppen	94
❖	Zend- en wisknop: submit, reset	94
❖	Knop: button	94
❖	Figuurknop: image	94
11.7	Structuur in het formulier brengen	95
❖	Fieldset en legend	95
11.8	CGI-scripts	96
❖	Een voorbeeld	96
❖	Verwerking van de inhoud van de formulierinvoer	97
11.9	Toegankelijk formulier	97
❖	Labels en invulvelden (input) zijn met elkaar verbonden:	97
❖	Input zonder label	97
❖	Label bij datumvelden	98
❖	Samenhorende formulervelden zijn ingesloten door een fieldset.	98
❖	Verplichte velden in label duidelijk maken	98
❖	Een duidelijke verzendknop	98
❖	Bij validatie je foutboodschap aanpassen	98
❖	Voorbeelden toegankelijkheid:	98
12	JavaScript	99
12.1	Wat is Javascript?	99
12.2	De <script> tags en je eerste javascript	100
12.3	Lexicale structuur	101
❖	Hoofdletters	101
❖	Scheiden van statements	101
❖	Commentaar	101
❖	Vorbehouden woorden	102
❖	Strict mode	102
12.4	Variabelen	103
❖	Getallen	103
❖	Wiskundig operatoren	103
❖	Toekenningoperatoren	103
❖	Speciale getallen	104
❖	String	105
❖	boolean	106
❖	Vergelijkende operatoren	106
❖	Logische operatoren	107
❖	null	107
❖	Undefined	107
❖	Zwakke typering	108
❖	Bereik van een variabele	108
❖	Let en const (ES6)	109
❖	Arrays	110
12.5	Controlestructuren	111
❖	if-else	111
❖	switch	112
❖	while	112
❖	do-while	113

❖	for.....	113
12.6	Functies	114
❖	Functie expressie en functie declaratie	114
❖	Geneste functies.....	114
❖	Immediately-Invoked Function Expression (IIFE)	115
❖	Functie met return-waarde functie.....	116
12.7	Objecten.....	117
❖	Wat is een Object?.....	117
❖	Objecten maken	118
❖	Properties en methodes.....	118
13	Document Object Model.....	118
13.1	Browser Object Model (BOM)	118
13.2	Document Object Model (DOM).....	119
13.3	W3C DOM.....	119
14	DOM scripting.....	125
14.1	Inleiding: Wat is DOM-scripting?	125
14.2	Event handling	125
❖	Event handling via JavaScript	125
❖	Event handlers voor documenten en vensters	126
❖	Event handlers voor alle elementen.....	126
❖	Event handlers voor formulieren	126
14.3	Pagina's dynamisch maken	127
❖	Methods en properties van window.....	127
❖	pop-upvensters: alert, prompt, confirm	129
14.4	DOM-scripting	130
❖	Benaderen van eigenschappen (properties).....	130
❖	Het gebruik van DOM Methods	131
15	Bronnen	132
16	jQuery	133
16.1	Inleiding	133
16.2	Selectors.....	134
❖	De basis selectors	134
❖	De selectors voor attributen	135
❖	Selectors voor formulieren.....	135
❖	Selectors op child of type.....	136
❖	Andere selectors.....	136
16.3	Verschil met JavaScript.....	137
❖	Selectie van één element.....	137
❖	Selectie van een reeks elementen.....	137
16.4	Interactie met de DOM-boomstructuur.....	138
❖	De methode .html	138
❖	De methode .val.....	138
❖	De methode .css	138
❖	De methode .attr en .prop.....	138
❖	De methode .append en appendTo	139
❖	De methode .prepend en .prependTo.....	139
❖	De methode .addClass, .hasClass, .removeClass en .toggleClass.....	140
❖	Verschillende methodes in een ketting.....	140
16.5	Event handling	141
16.6	Animaties	143

1 Inleiding HTML5

1.1 Markeertalen en webstandaarden

Om websites te kunnen bezoeken moeten de computers die aangesloten zijn op het internet bepaalde protocollen hanteren. Deze protocollen zijn beschreven in een aantal standaarden, door o.a. het **World Wide Web Consortium** (W3C), de Internet Engineering Task Force (IETF), Ecma International en de Web Hypertext Application Technology Working Group (WHATWG).

Het protocol dat voor deze cursus belangrijk is het **HyperText Transfer Protocol** (http) dat platformafhankelijk werkt zodat het er niet toe doet welk besturingssysteem aan elke zijde van de communicatie gebruikt wordt.

Voor de ontwikkeling van websites voor het World Wide Web worden **markeertalen** gebruikt. Markeertalen zijn niets meer dan een set van regels die toelaten om de opmaak, het formaat en de structuur van tekst binnen een document te bepalen. De markeertaal **Standard Generalized Markup Language** (SGML) kan als voorloper voor HTML, XML, XHTML en HTML5 beschouwd worden. Bij de opkomst van het internet is een lichtere versie van SGML ontwikkeld, nl. HTML (**Hypertext Markup Language**). De HTML-standaard is telkens uitgebreid tot de huidige HTML-specificatie. Ten tijde van HTML 3.2 voegden vele browserontwikkelaars eigen elementen toe die echter niet langer compatibel zijn met andere browsers. Met de komst van HTML 4 wou het W3C een einde maken aan deze “browseroorlog” door één standaard te bepalen die compatibel is met alle browsers. Vanaf HTML 4 komt er een scheiding tussen de taal die de structuur bepaalt (HTML) en de taal die de presentatie bepaalt (CSS). In 1999 is de HTML 4.01 standaard beschreven met een aantal aanpassingen op de vorige standaard om zo een nog uniformere standaard te bekomen.

De beperkte grootte, het gelimiteerd aantal elementen en het gebruiksgemak zijn de sterke punten van HTML maar beperken eveneens de verdere ontwikkeling. Door de beperkingen van HTML is XHTML 1.0 in 2000 ontwikkeld als opvolger van HTML 4.01. XHTML (**Extensible Hypertext Markup Language**) is het resultaat van de hervorming van HTML 4 in XML 1.0 (**Extensible Markup Language**). XHTML behoudt dus de elementen van HTML 4.01 maar wordt geschreven als XML. Dit zorgt voor een meer rigide taal die “forward compatible” is.

In 2004 werd de WHATWG opgericht, een initiatief van Apple, Mozilla en Opera. Zij ontwikkelde een opvolger voor HTML 4. Gelijktijdig werkte het W3C verder aan de opvolger van XHTML (XHTML 2.0). In 2007 stopt het W3C met deze ontwikkeling en neemt de voorstellen over van de WHATWG in een eerste draft van de nieuwe markeertaal met de officiële benaming HTML5 (zonder spatie).

Hoe bepaalt een browser nu wat voor pagina's worden aangeboden in deze wirwar van standaarden? Om compatibiliteitsredenen zal een browser er vanuit gaan dat er “oude HTML” wordt aangeboden, zodat de voordelen van “nieuwe HTML” eigenlijk verloren gaan. Om dit te voorkomen is het beter de browser mee te delen welke standaard in het document gebruikt wordt, zodat er geen twijfel kan bestaan hoe alles geïnterpreteerd dient te worden.

❖ HTML5

HTML5 is sinds 28 oktober 2014 een W3C-aanbeveling. Een vernieuwde versie, HTML 5.1, is op 1 november 2016 gepromoveerd tot aanbeveling. Naar verwachting zal eind 2017 HTML 5.2 worden gefinaliseerd. Ondertussen houdt WHATWG de HTML-specificatie continu up-to-date, als een ‘levende standaard’. HTML5 heeft de laatste jaren dus nog wel wat wijzigingen ondergaan. Webpagina's kunnen dus binnen deze standaard worden ontwikkeld, maar het blijft raadzaam om de ondersteuning door de verschillende browsers te controleren.

Door de voorgeschiedenis van HTML5 (HTML 4 en XHTML 2) worden er twee verschillende syntaxen ondersteund: de losse HTML-syntax en de strikte XML-syntax. De HTML-syntax wordt veruit het meest gebruikt.

◆ *HTML-syntax*

De browser herkent HTML5 aan de hand van een document type (DOCTYPE) declaratie. Deze DOCTYPE komt helemaal bovenaan in het document. In vroegere HTML-versies bevatte zo'n DOCTYPE een verwijzing naar een document type definition (DTD), die o.a. aangeeft wat de toegelaten elementen zijn. In HTML5 is hiervan afgestapt, omdat browsers deze DTD in feite niet bekijken.

De overblijvende DOCTYPE is zeer eenvoudig:

```
<!DOCTYPE html >
```

Deze DOCTYPE is noodzakelijk! Anders veronderstellen browsers dat de code gemaakt is voor oudere browsers, waardoor ze diverse bugs van toen nabootsen (quirks-mode).

◆ *XML-syntax*

De browser herkent HTML met XML-syntax (XHTML) aan de hand van de Content-Type header. Deze header wordt door de server gebruikt om het type resource, het MIME-type, aan te geven. Bij bijv. een PNG-afbeelding is dit image/png. Voor XHTML is het application/xhtml+xml.

De volgende header moet dus verzonden worden door de server (niet in de code):

```
Content-Type: application/xhtml+xml
```

Deze header is noodzakelijk! Anders veronderstellen browsers dat je code met html-syntax doorstuurt.

Verder is het `xmlns`-attribuut van het openingselement verplicht voor alle XML-pagina's. De waarde van dit attribuut bepaalt de namespace waartoe de gebruikte elementnamen van het document behoren. Voor XHTML geeft dit:

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

De eerste regel van een XML-document bevat de optionele XML-proloog die beschrijft welke versie van XML in het document gebruikt wordt. Deze declaratie staat in een `<?xml ...?>` processing instruction en kan drie attributen bevatten: `version`, `encoding` en `standalone`.

```
<?xml version="..." encoding="..." standalone="..."?>
```

- `version` = staat op "1.0" of "1.1" en geeft de gebruikte versie weer. Begin 2017 ondersteunen browsers versie 1.1 niet. Enkel versie 1.0 is dus beschikbaar.
- `encoding` = dit is optioneel en beschrijft de karakterset. Wanneer dit attribuut wordt weggelaten, geldt UTF-8 als karakterset.
- `standalone` = dit is optioneel en kan op "yes" of "no" (standaard) gezet worden. De `standalone` geeft aan of het document alle informatie zelf bevat of aangewezen is op externe DTD's voor zijn declaratie.

De DOCTYPE is hetzelfde als bij de HTML-syntax en is optioneel bij HTML5 in XML-syntax.

◆ *Syntax in deze cursus*

Bij de introductie van XHTML hekelden vele webontwikkelaars de strenge benadering terwijl andere de rigide correctheid toejuichten. In deze cursus wordt geopteerd voor de HTML-syntax, aangevuld met code conventies gebaseerd op de XML-syntax. Hierdoor is de code consistent en goed leesbaar.

◆ *HTML-element*

Het openingselement `<html>` bevat verder het optionele, aangeraden `lang`-attribuut. Dit is o.a. belangrijk als de pagina door screenreaders gelezen wordt om de uitspraak te bepalen.

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

◆ *Meta-informatie*

Om correct te kunnen valideren en ervoor te zorgen dat de browser weet hoe hij de inhoud moet interpreteren en weergeven, dient bij de HTML-syntax een `meta`-tag toegevoegd te worden die aangeeft welke karakterset gebruikt wordt, de `charset`.

```
<meta charset="utf-8">
```

Ontleding van bovenstaande regel:

- `meta`: de informatie is bedoeld voor de browser
- `charset`: geeft aan dat er tekst volgt die als HTML dient geïnterpreteerd te worden en die opgebouwd is met de utf-8 karakterencodering. De mogelijke karaktersets zijn ISO-8859 en UTF (Unicode).

De ISO-8859 standaarden zijn:

- ISO-8859-1 of Latin-1 = West-Europese talen (Frans, Spaans, Catalaans, Baskisch, Portugees, Italiaans, Albanees, Nederlands, Duits, Deens, Zweeds, Noors, Fins, Faroese, IJslands, Iers, Schots en Engels).
- ISO-8859-2 of Latin-2 = Centraal en Oost-Europese talen (Tsjechisch, Hongaars, Roemeens, Kroatisch, Slowaaks, Sloveens, Servisch)
- ISO-8859-3 of Latin-3 = Zuid-Europese talen (Esperanto, Maltees, ..)
- ISO-8859-4 of Latin-4 = Baltische staten (Estlands, Litouws, Lets, Groenlands, Laplands).
- ISO-8859-5 of Cyrillic = talen met cyrillisch schrift (Bulgaars, Macedonisch, Russisch).
- ISO-8859-6 of Arabic = Arabische talen
- ISO-8859-7 of Greek = Grieks
- ISO-8859-8 of Hebrew = Hebreeuws en Yiddisch
- ISO-8859-9 of Latin-5 = Turks
- ISO-8859-11 of Thai = Thaise taal
- ISO-8859-15 of Latin-9 = Update van Latin-1.

Daarnaast kan de gebruikte karakterset ook Unicode zijn, dit wordt aangegeven met UTF-8, UTF-16 of UTF-32. De 16-bit en 32-bit versies zijn enkel nodig wanneer veel Oosterse tekens dienen weergegeven te worden en een kleinere bestandsgrootte wenselijk is.

Vuistregel: gebruik utf-8 maar zorg dat je bestanden effectief geëncodeerd zijn met utf-8.

◆ *Afgekeurde elementen*

Heel wat elementen van de vroegere HTML-versies zijn afgekeurd bij de overgang naar XHTML. De meeste van die elementen blijven afgekeurd of sterk afgeraden in HTML5. Het is de bedoeling om inhoud en opmaak te scheiden. Het zijn dus de elementen en attributen die puur opmaak beschrijven en geen betekenis geven aan de inhoud die zijn verdwenen. Als je deze maatstaf gebruikt is het gemakkelijker te onthouden welke attributen "deprecated" zijn of niet.

Voor de volledigheid is in de bijlagen een lijst opgenomen van verschillende geldige en afgekeurde elementen en de HTML-versie waartoe ze behoren.

1.2 Basis van een webpagina

❖ Structuur

HTML-webpagina's hebben eenzelfde structuur waarbij ze opgedeeld zijn in een hoofd (head) en een lichaam (body). Het belangrijkste verschil in de structuur situeert zich in de HTML-hoofding.

- HTML5-pagina

UTF-8 code:

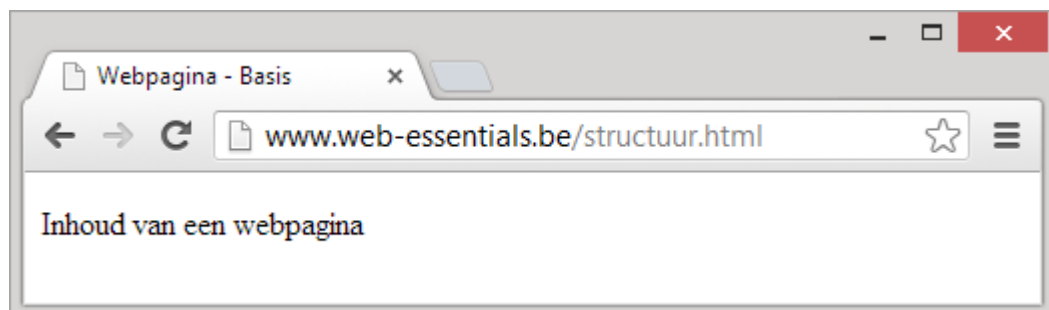
```
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="utf-8">
  <title>Webpagina - Basis</title>
</head>
<body>
  <p>Inhoud van een webpagina</p>
</body>
</html>
```

- HTML5-pagina (XML-syntax)

Header: Content-Type: application/xhtml+xml

UTF-8 code:

```
<html xmlns="http://www.w3.org/1999/xhtml" lang="nl">
<head>
  <title>Webpagina - Basis</title>
</head>
<body>
  <p>Inhoud van een webpagina</p>
</body>
</html>
```



❖ Verklaring van de opbouw

♦ *Het <html>-element*

De `<html>...</html>` container geeft aan dat het om een HTML-document gaat. De volledige inhoud van het webdocument valt tussen deze tags.

♦ *Het <head>-element*

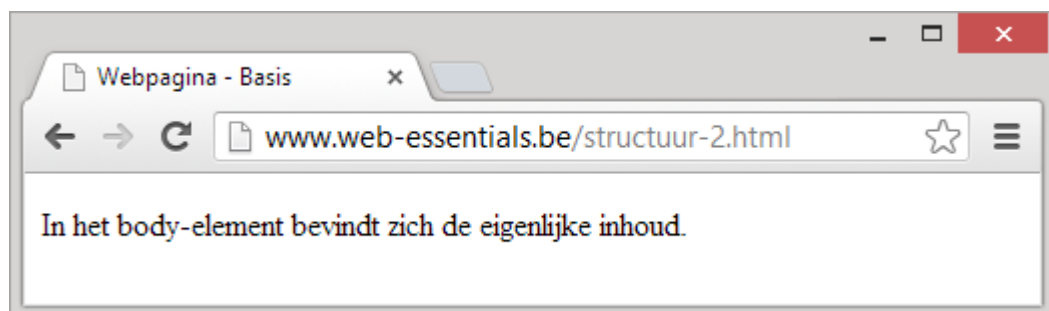
De hoofding `<head>...</head>` bevat informatie die betrekking heeft op het document, maar die niet in het documentvenster wordt weergegeven. In de `<head>` kan je slechts een beperkt aantal elementen gebruiken:

- `<title>...</title>`: bevat de verplichte titel van een webpagina. Er is slechts één titel per webdocument mogelijk en die dient zo beknopt en verklarend mogelijk te zijn. Je mag ook geen opmaak of speciale tekens gebruiken.
- `<base />`: specificeert een absoluut basisadres (`href`-attribuut) en doel (`target`-attribuut).
- `<link />`: definieert de relatie van het document met andere documenten en wordt vooral gebruikt om naar externe stijlbestanden te verwijzen (zie CSS).
- `<meta />`: additionele informatie over het document. Dit bevat instructies voor browsers en zoekmachines en het inhoudstype.
- `<style>...</style>`: voor de definitie van stylesheets die binnen het document zelf worden toegepast (afgeraden, zie CSS).
- `<script>...</script>`: bevat scripts of legt een link met scripts (zie JavaScript).

♦ *Het <body>-element*

Al de verdere tekst, afbeeldingen en codes bevinden zich binnen de `<body>...</body>` container. Alle opmaak-attributen voor het `<body>`-element zijn afgekeurd en dien je te vervangen door stylesheets.

```
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="utf-8">
  <title>Webpagina - Basis</title>
</head>
<body>
  <p>In het body-element bevindt zich de eigenlijke inhoud.</p>
</body>
</html>
```



1.3 Elementen, tags en attributen

❖ Elementen en hun tags

HTML-**elementen** zijn de instructies die de opmaak van de tekst, de figuren, de tabellen, de hyperlinks, enzovoort bepalen. Een **tag** is strikt genomen een onderdeel van een HTML-element. Voor de meeste elementen is er een open- en een sluittag, dit noemt men **containertags**. Slechts enkele elementen zijn **open** of leeg en worden dus door slechts één tag aangegeven.

◆ *Containertag:*

Deze tags beïnvloeden de tekst die tussen de open- en de sluittag staat. De combinatie van de opentag, de inhoud en sluittag vormt het HTML-element. Een voorbeeld is `<body>...</body>`.

◆ *Lege tag:*

Deze tags gebieden de browser eenmalig actie te ondernemen om een bepaald effect te bekomen. Aangezien in XHTML alle elementen dienen afgesloten te worden met een sluittag worden de open HTML-elementen geschreven met een "/" op het einde. Binnen HTML5 is het niet langer noodzakelijk om de tag af te sluiten. Voorbeelden zijn `
` en ``.

❖ Attributen

Haast alle HTML-elementen kan je een aantal extra kenmerken meegeven door middel van attributen. Ze vergroten de functionaliteit van de elementen. Het attribuut staat steeds in de open tag. Indien meerdere attributen worden opgegeven speelt de volgorde geen rol.

Aan het correct gebruik van attributen zijn niet langer voorwaarden verbonden. In deze cursus hanteren we echter de volgende regels:

- Alle namen staan in kleine letters;
- De waarde die meegegeven wordt aan de attributen staat tussen dubbele aanhalingstekens;

```

<a href="page2. html ">...</a>
<style type="text/css">...</style>
```

De HTML5-standaard heeft enkele kernattributen die je in haast elk element kan gebruiken.

kernattribuut	waarde
accesskey	snelkoppeling om een element te benaderen.
class	klasse waartoe het element behoort.
contenteditable	maakt de inhoud aanpasbaar door de gebruiker.
dir	de tekstrichting.
draggable	maakt het element versleepbaar.
dropzone	beschrijft de actie als een item in een element gedropt wordt.
hidden	maakt het element verborgen.
id	unieke identificatie die in het hele document geldt.
lang	bepaalt de taal van de elementinhoud.
spellcheck	bepaalt of de spellingscontrole moet toegepast worden.
style	bijhorende informatie over de opmaakstijl.
tabindex	bepaalt de tabvolgorde van het element.
title	titel van het element.

De elementen waarbij deze kernattributen niet toegepast kunnen worden, zijn `<base>`, `<head>`, `<html>` (uitgezonderd het `lang`-attribuut), `<meta>`, `<param>`, `<script>`, `<style>` en `<title>`.

❖ Commentaar

Tussen commentaartags kan informatie geplaatst worden die genegeerd wordt door de browser. Net zoals in programmeertalen biedt het plaatsen van commentaar samen met inspringen een verhoogde leesbaarheid. De commentaar kan enkel in de broncode bekeken worden.

```
<!-- Commentaar -->
```

1.4 Verschil tussen de verschillende HTML5-syntaxen

Ondanks de bijna volledige overlap van elementen en attributen in de HTML- en de XML-syntax zijn er toch belangrijke verschillen die hier nogmaals worden opgesomd.

De XML-syntax:

- In XML-documenten is optioneel een XML-hoofding aanwezig;
- XML heeft een strikter formaat; elke fout wordt getoond aan de eindgebruiker;
- De strikte XML-syntax is eenvoudig te verwerken door niet-HTML tools;
- Alle elementen en attributen moeten in kleine letters ingegeven worden. In tegenstelling tot HTML is XHTML hoofdlettergevoelig (bv. `<table>` en niet `<Table>` of `<TABLE>`);
- Alle elementen moeten een eindtag bezitten (bv. `
`, ``);
- Alle attribuutwaarden staan tussen apostrofs of aanhalingstekens. Je hebt de vrije keuze tussen dubbele of enkele, maar je mag ze niet onderling mengen bij een waarde (bv. `width="50%"` of `width=' 50%'`, maar niet `width=' 50%"` of `width="50%'`);
- Verkleinde attributen zijn niet langer toegelaten, maar moeten als attribuut="attribuutwaarde" geschreven worden (bv. `selected="selected"` i.p.v. `selected`);
- Alle inhoud moet binnen een tag opgenomen worden. Losse tekst of andere inline elementen rechtstreeks ingeven in de body is niet toegelaten, maar moet je in een paragraaf of een ander containerelement steken;
- Bij geneste elementen mag er geen overlapping zijn tussen de elementen (bv. `<i>Cursief en vet</i>` en niet `<i>Cursief en vet</i>`).

De HTML-syntax is veel flexibeler. De regels waar je je nog moet aan houden zijn:

- Plaatsing van de eenvoudige DOCTYPE-declaratie;
- Het strikte formaat van XML mag nog, maar hoeft niet;
- HTML5 is niet hoofdlettergevoelig (bv. `<table>` of `<TABLE>`);
- Open elementen mogen maar moeten geen eindtag bezitten (`` of ``);
- Het plaatsen van attributen tussen apostrofs of aanhalingstekens wordt optioneel, zolang er geen witruimte (spaties, ...) in voorkomt. (bv. `width="50%"` of `width=' 50%'` of `width=50%`);
- De verkleinde attributen zijn toegelaten (`selected`);
- Alle inhoud moet binnen een tag opgenomen worden;
- Het nesten van elementen blijft strikt.

Zoals eerder aangehaald, wordt in deze cursus geopteerd voor de HTML-syntax, aangevuld met code conventies gebaseerd op de XML-syntax. In de voorbeeldcodes en oefeningen worden altijd de volgende syntaxregels nagevolgd, ook al zijn de meesten niet noodzakelijk om geldige HTML5-code te schrijven.

- UTF-8 tekst;
- <DOCTYPE html> en <meta charset="utf-8"> aanwezig;
- Elementen en attributen altijd in kleine letters;
- Attributen worden tussen aanhalingstekens geplaatst;
- Eén indentatieniveau per open tag aan het begin van de regel, behalve <html>;
- Geen opmaak-elementen, tenzij specifiek voor het voorbeeld;

De volgende HTML-code moet dus aangepast worden om een geldig HTML5-document te krijgen.

```
<HTML>
<Head>
  <title>Web Technology</title>
</head>
<BODY>
<h1>Vak: <i>Web Technology</i></h1></i>
<img SRC=astridrm.gif Alt="Astrid Ramakers">
<p>De volgende onderdelen worden behandeld:
<UL>
<li>HTML5
<li>CSS
  <li>DOM
<li>JavaScript
  </li>
</UL>
</BODY>
</html>
```

De correcte HTML5-code is:

```
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="utf-8">
  <title>Web Technology</title>
</head>
<body>
  <h1>Vak: <span class="vak">Web Technology</span></h1>
  <p></p>
  <p>De volgende onderdelen worden behandeld: </p>
  <ul>
    <li>HTML5</li>
    <li>CSS</li>
    <li>DOM</li>
    <li>JavaScript</li>
  </ul>
</body>
</html>
```

Browsers zijn zeer vergevingsgezind. Hierdoor leveren inbreuken tegen het gebruik van kleine letters en het plaatsen van aanhalingstekens niet onmiddellijk problemen op bij de weergave in de browser. Bovenstaande codes geven eenzelfde resultaat in een browser, ook al zit de pagina vol fouten.

1.5 Validatie van de code

Ondanks deze grote vergevingsgezindheid van de meeste browsers, zorg je er toch best voor dat je webpagina's voldoen aan de standaard om zo het meest kans te hebben dat ze in alle browsers identiek worden weergegeven. Deze controle ten opzichte van de standaard heet **validatie**. Bij de validatie wordt een HTML5-document gecheckt ten opzichte van de vereisten in de specificatie.

Niet enkel zal er gecontroleerd worden of de elementen en attributen bestaan, maar eveneens zullen alle andere fouten aangeduid worden. De andere veel voorkomende fouten zijn:

- Containerelementen niet afsluiten;
- Foutief nesten van elementen;
- Het opnemen van losse tekst (niet in een container);
- Schrijffouten (bijv. scr i.p.v. src);
- Het vergeten afsluiten van apostroffen en aanhalingstekens;
- Het gebruiken van afgekeurde elementen en attributen;
- En nog vele andere

Om te valideren kan je gebruik maken van ingebouwde validators in een HTML-editor of van webbased validators. De bekendste hiervan is <http://validator.w3.org>. Bij deze validator van het World Wide Web Consortium (W3C) kan je zowel code als online en offline documenten controleren.

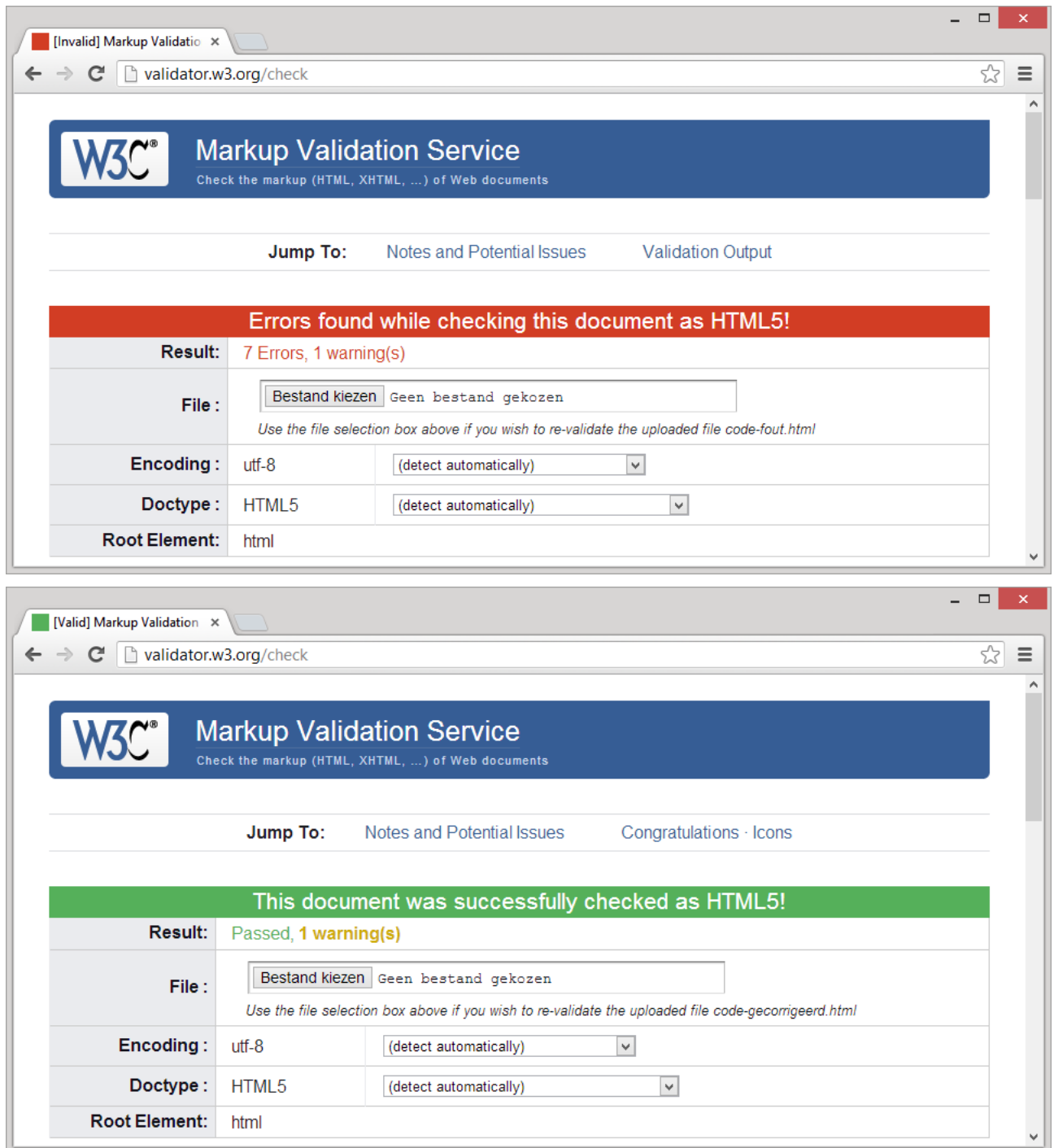
Het controleren van een onderstaande code geeft 7 fouten weer in het validatorvenster. Dit aantal fouten komt niet altijd overeen met het echte aantal aangezien het vergeten van een sluittag of een aanhalingsteken zich kan doorrekenen in de rest van het document en zo meer fouten genereert. Het is dus belangrijk om de fouten van het begin naar het einde van het document te verbeteren.

```
<!DOCTYPE html >
<html >
<head>
  <meta charset="utf-8">
  <title>Val i dat i epagi na</ti tle>
</head>
<body>
  <h1>Vak: <span class="vak">Web Technol ogy</h1></span>
  <hr width="30%">
  De volgende onderdel en worden behandel d:
  <ul>
    <li>HTML</li>
    <li>CSS</li>
    <li>JavaScri pt</li>
  <body>
</body>
</html >
```

Na correctie van de code krijg je wel een gevalideerd document.

```
<!DOCTYPE html >
<html >
<head>
  <meta charset="utf-8">
  <title>Val idati epagi na</title>
  <link rel="stylesheet" href="style.e.css">
</head>
<body>
  <h1>Vak: <span class="vak">Web Technol ogy</span></h1>
  <hr>
  <p>De volgende onderdel en worden behandel d: </p>
  <ul>
    <li>HTML5</li>
    <li>CSS</li>
    <li>JavaScri pt</li>
  </ul>
```

```
</body>
</html>
```



1.6 Browsers

Er zijn zeer veel browsers beschikbaar om HTML-pagina's mee te bekijken. Onderhuids kunnen ze opgedeeld worden in vier types van lay-outengine (Trident, EdgeHTML, Gecko en Blink/WebKit). Al deze engines en de op hun gebaseerde browsers hebben andere eigenschappen en verschillen ook in de ondersteuning van de verschillende webstandaarden.

❖ De vier browserengines

◆ *Trident*

Microsoft beheert de Trident-engine en gebruikt die enkel voor Windows-toepassingen. De meest gekende browsers en toepassingen zijn Internet Explorer (IE4-IE11) en Microsoft Outlook. Het is een gesloten standaard, dus met licentiekosten.

In 2002 had Microsoft bijna een monopolie met een marktverdeling van ongeveer 85%. De enige concurrenten waren toen AOL en Netscape. Momenteel dreigt IE onder de 10% te komen. De late ondersteuning van de verschillende webstandaarden en de vele pluginproblemen zouden daarvan de oorzaak kunnen zijn.

◆ *EdgeHTML*

De EdgeHTML engine is gemaakt door Microsoft voor de Edge browser, als antwoord op de problemen die Trident meesleurt uit het verleden. De engine ondersteunt bijv. oude programma's die voor een deel oude html-code willen tonen niet. Dit vermindert de complexiteit, waardoor Microsoft sneller nieuwe standaarden kan implementeren en de engine sneller, stabiel en veiliger is dan Trident. Internet Explorer en Trident worden dan ook niet verder ontwikkeld. Edge heeft momenteel een marktaandeel van ongeveer 5%.

◆ *Gecko*

De populaire op de Gecko-engine gebaseerde toepassingen zijn Firefox, Thunderbird en vroeger Netscape. Dit zijn allemaal open-source pakketten die beschikbaar zijn voor Windows, Mac OS X, en Linux/BSD. De ontwikkeling is in 1997 gestart met Netscape, maar is nu overgenomen door de Mozilla Corporation. Firefox heeft de laatste jaren veel marktaandeel verloren aan Chrome.

◆ *Blink/WebKit*

De open-source webkit-engine wordt gebruikt door de WebKit Foundation, Apple, Adobe, en het KDteam en is beschikbaar voor alle grote besturingssystemen, inclusief iOS voor de iPhone en iPod Touch. Blink is een fork van deze engine door Google, maar beiden blijven grotendeels dezelfde ondersteuning bieden. De bekende toepassingen zijn Safari, Chrome, Epiphany (Gnome) en Konqueror (KDE). Binnen Android is een Chrome-browser voorzien. De huidige marktaandelen van Safari en Chrome zijn respectievelijk rond de 20% en 45% in België. Sinds Google pc-bouwers betaalt om Chrome voorgeïnstalleerd mee te leveren bij nieuwe pc's, is het voor concurrenten veel moeilijker geworden om de gewone gebruiker te overtuigen hun browser(s) te gebruiken.

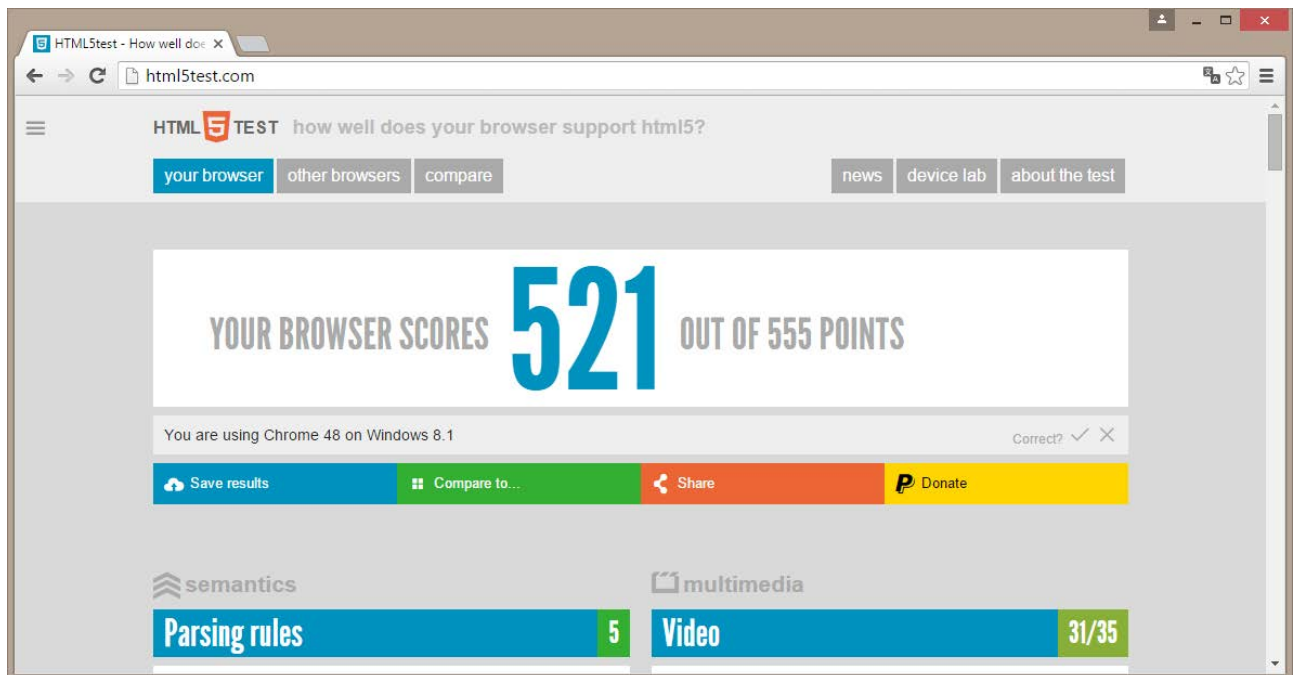
Alle browserontwikkelaars hebben de laatste jaren ook versies voor mobiele telefoons en tablets toegevoegd aan hun aanbod.

❖ HTML5-ondersteuning

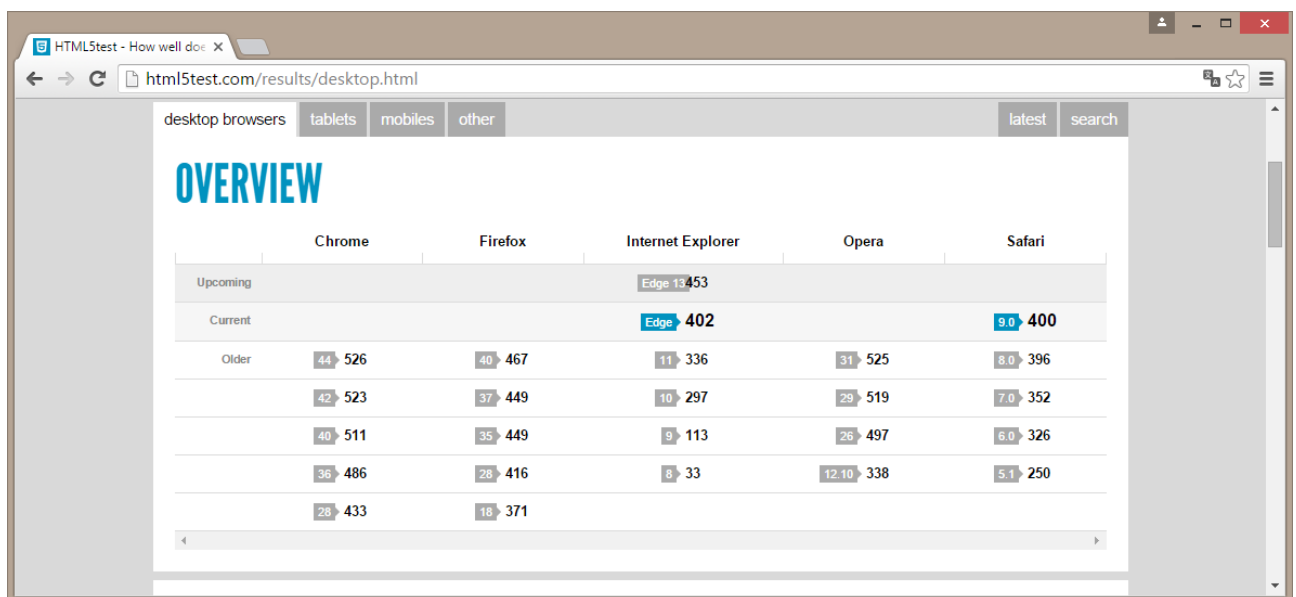
De HTML5-standaard is pas sinds eind 2014 een aanbevolen W3C-standaard. Voor 2010 was er een tijdlang weinig vernieuwing qua webtechnologie, maar alle browsers zijn aan een heuse inhaalbeweging bezig. De versies van de browsers volgen elkaar steeds sneller op.

Vooraleer nieuwe HTML5-elementen te gebruiken in je webpagina, test je dus best de ondersteuning door de meest gebruikte browsers en meer bepaald in de versies die door jouw eindgebruikers het meest gebruikt worden. Bovendien gebruiken steeds meer mensen mobiele browsers, die soms andere zaken ondersteunen dan hun gelijknamige pc-aanverwanten.

Je kan een gedetailleerde score per browser opvragen om <http://www.html5test.com>. Hou er echter rekening mee dat sommige browsers bewust zaken niet ondersteunen, o.a. omwille van privacy (bijv. ping-attribuut van <a>). Onderstaande screenshot geeft de score voor Chrome versie 48.



Op de tab "other browsers" kan je de score opvragen van de browsers voor desktops, tablets en mobiele telefoons.



Andere nuttige websites om de ondersteuning van HTML5 te controleren zijn:

- <http://www.html5doctor.com>
- <http://www.caniuse.com>

Op de volgende website kan je meer informatie vinden over een template waarbij de ondersteuning voor oudere browsers wordt opgevangen.

- <http://www.html5boilerplate.com>

2 Structuur

Alle gewone tekst, geplaatst binnen het `<body>`-element, wordt door de browser rechtstreeks op het scherm geplaatst, maar alle in een tekstverwerker toegevoegde tekstopmaak zoals returns, meervoudige spaties en tab's worden door de browser genegeerd. Tekst die als tabel opgemaakt is in een editor wordt dus volledig op een hoop gegooid. Om structuur en semantische betekenis aan een tekst mee te geven kan gebruik gemaakt worden van een aantal structuurelementen.

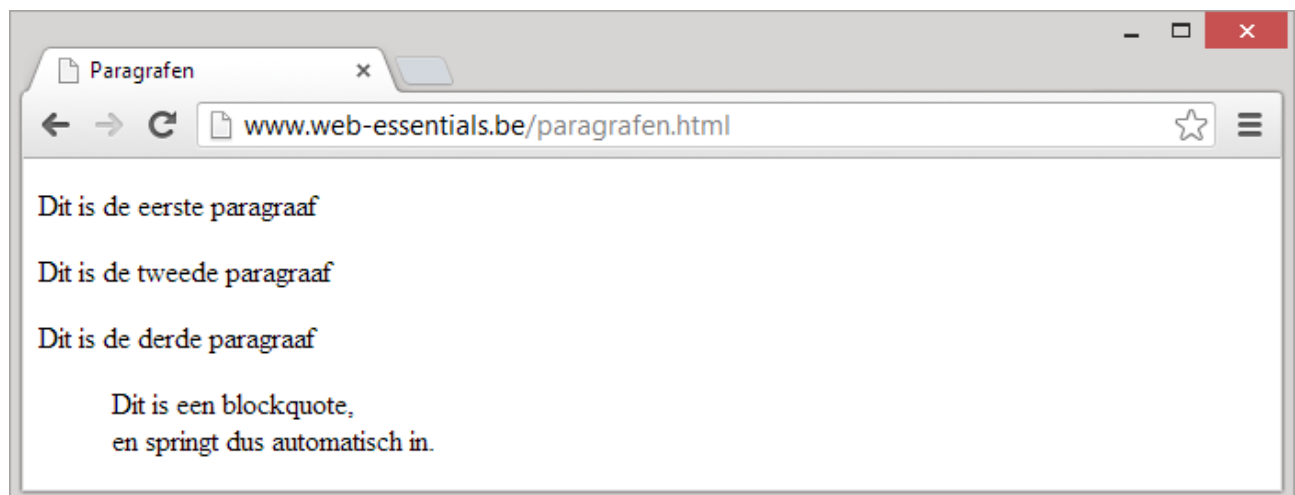
2.1 Blokelementen

Sommige HTML-elementen zijn **blokelementen** en worden altijd gescheiden van de andere elementen door de overgang naar een nieuwe regel. Blokelementen worden dus standaard onder elkaar getoond en niet naast elkaar. Dit geldt voor de paragraaf `<p>`, de `<blockquote>`, de koppen `<h1>` tot en met `<h6>`, de dividers `<div>`, de `<pre>`, alle elementen van lijsten (`ul`, `ol`, `li`, `dl`, `dt`, `dd`) en tabellen (`table`, `tr`, `td`, `th`), ... Met CSS kan daarna elk blokelement een opmaak meekrijgen die geldt voor de volledige inhoud, bijvoorbeeld de horizontale uitlijning (left, right, center en justify).

❖ Paragrafen

De paragraaf `<p>...</p>` dient om een alinea aan te geven. De browser voegt in de standaard layout een lege regel toe zodat afgescheiden alinea's of paragrafen bekomen worden. In auditieve browsers wordt een pauze gelaten tussen het voorlezen van twee paragrafen. In een paragraaf kan nooit een ander blokelement opgenomen worden (zoals koppen, andere paragrafen, enzovoort), maar mogen wel inline-elementen genest worden (zoals ``, ``, ...).

```
<p>Dit is de eerste paragraaf</p>
<p>Dit is de tweede paragraaf</p>
<p>Dit is de derde paragraaf</p>
```



❖ Blockquote

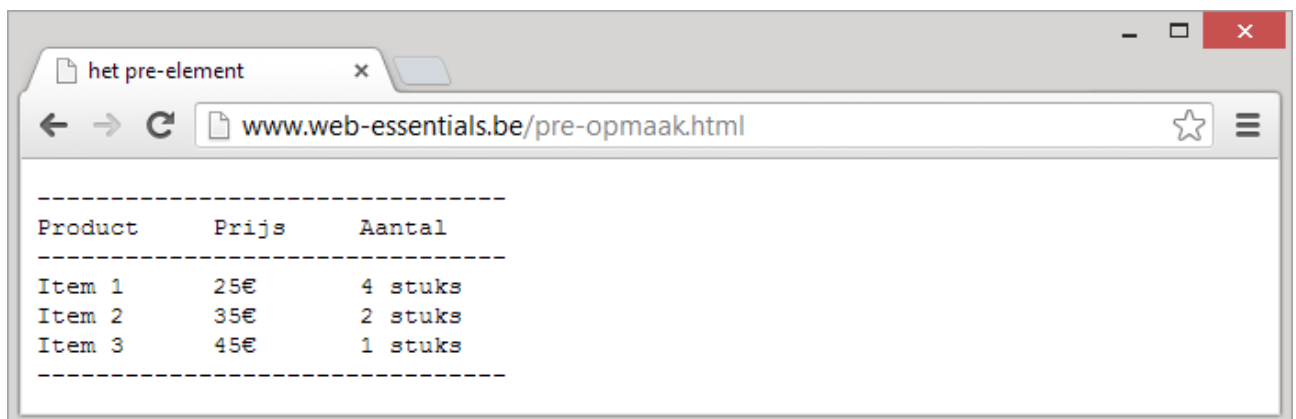
De blockquote `<blockquote>...</blockquote>` was oorspronkelijk bedoeld om te refereren naar beroemde uitspraken. Nu stelt het een citaat uit een andere bron voor. Kenmerkend is het automatisch inspringen en de mogelijkheid om andere blokelementen te kunnen nesten.

```
<blockquote>
  <p>Dit is een blockquote, <br>
    en springt dus automatisch in. </p>
</blockquote>
```

❖ Preformatted

Met `<pre>...</pre>` wordt een voorgevormde (preformatted) uitlijning overgenomen door de browser. De tekst wordt dus overgenomen, inclusief tab's en meervoudige spaties. Het nadeel van dit element is dat de betekenis die de tekststructuur geeft aan de inhoud zonder extra elementen niet kan begrepen worden door o.a. zoekmachines en niet-visuele browsers.

```
<pre>
-----
Product      Prijs      Aantal
-----
Item 1       25 EURO   4 stuks
Item 2       35 EURO   2 stuks
Item 3       45 EURO   1 stuks
-----
</pre>
```



❖ Koppen

Binnen HTML zijn zes koppen (headings) gedefinieerd. Koppen zijn blokelementen en nemen dus telkens een nieuwe alinea in. Startend met `<h1>...</h1>` tot en met `<h6>...</h6>` worden telkens kleinere koppen gegenereerd. De normale tekstgrootte situeert zich meestal tussen `<h3>` en `<h4>`.

```
<body>
<h1>Koptitel 1</h1>
<h2>Koptitel 2</h2>
<h3>Koptitel 3</h3>
<h4>Koptitel 4</h4>
<h5>Koptitel 5</h5>
<h6>Koptitel 6</h6>
<p>gewone tekst</p>
</body>
```



❖ Adres

De tag `<address>...</address>` is ook een blokelement en wordt gebruikt om contactinformatie te tonen die verband houdt met het dichtstbijzijnde bovenliggende `<article>`- of `<body>`-element. Het `address`-element dient niet gebruikt te worden voor postadressen, tenzij die adressen in feite de relevante contactinformatie zijn. Je vindt dit vaak terug samen met andere informatie in een footer-element.

```
<address>Ken Barbie <a href="mailto:ken.barbie@x.com">mail me</a></address>
```

❖ Divisies

De tag `<div>...</div>` heeft geen betekenis. Ze kan echter elementen bevatten en groepeert zo deze kinderelementen. Met dit element kunnen groepen worden gedefinieerd om op te maken met CSS, zonder ze een eigen betekenis te moeten geven. Dit element kan enkel gebruikt worden als de groep effectief niets betekent, anders is een ander element op zijn plaats.

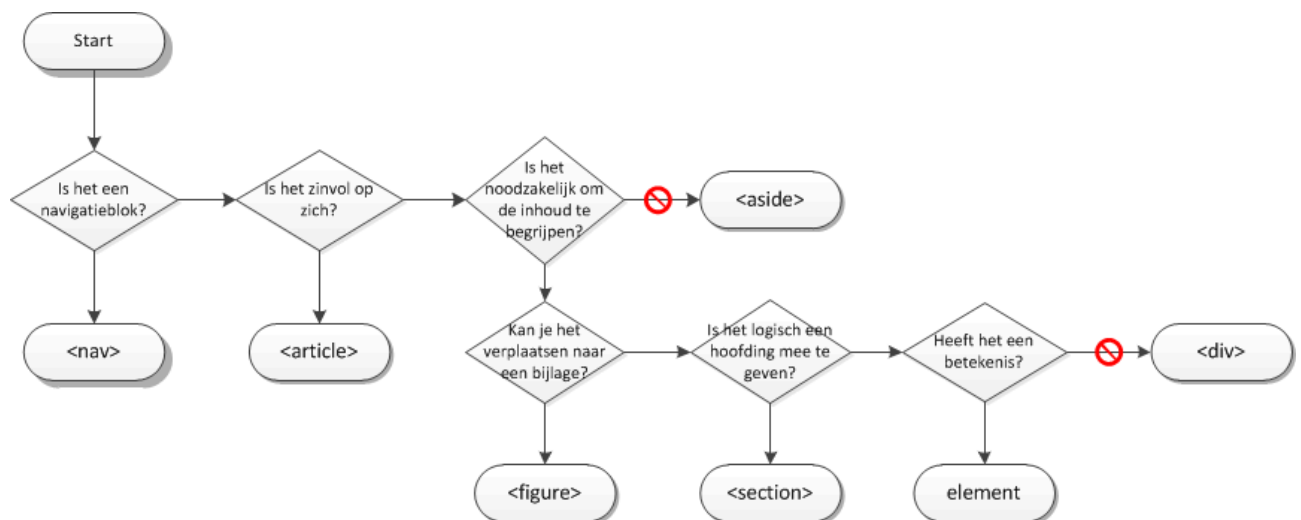
Dit element is onmisbaar in moderne webpagina's die uitvoerig gebruik maken van CSS. Deze divisies worden meestal samen gebruikt met stijldefinities om zo een lagenstructuur te bekomen (zie hoofdstuk over positioneren met CSS).

2.2 De structuurelementen van HTML5

In de voorlopers van HTML5 werd een nietsbetekenend blokelement toegevoegd, het `div`-element. Deze `div`'s zeggen echter niets over de inhoud waardoor het interpreteren van een pagina met veel `div`-elementen erg moeilijk is. In HTML5 zijn nieuwe structuurelementen toegevoegd om deze tekortkoming op te lossen. De zeven semantische blokelementen om de structuur van een webpagina te bepalen zijn `<section>`, `<main>`, `<article>`, `<header>`, `<footer>`, `<nav>` en `<aside>`.

Dit wil niet zeggen dat het `div`-element in onbruik is geraakt of fout is. Soms blijft het nuttig om bepaalde elementen te groeperen zonder daar een bepaalde betekenis aan te geven. In het geval dat de blok een structuur van de pagina beschrijft is het echter nodig gebruik te maken van de structuur elementen.

Het is niet altijd eenvoudig om te bepalen wanneer je moet kiezen voor welk structuurelement om een bepaalde inhoudsblok te definiëren. Vooral de keuze tussen `section`, `article` en `aside` is voor interpretatie vatbaar. Hiervoor bestaat geen perfecte regel, enkel een leidraad. Het is vooral belangrijk om consequent te zijn waar mogelijk en de voorbeelden in de standaarden te begrijpen. Een handige manier om consequent te blijven is de redenering volgen van de website HTML5 doctor (<http://html5doctor.com/downloads/h5d-sectioning-flowchart.pdf>). De onderstaande figuur is een vertaling van hun flowchart.



❖ Main

Het `main`-element kan gebruikt worden als groep voor de dominante inhoud van een ander element. Als het bovenliggende element veel kinderelementen telt, kan hiermee worden aangegeven wat de essentiële inhoud is. Tegenwoordig kan een pagina meerdere `main`-elementen bevatten, voor bijv. de belangrijkste inhoud van verschillende `article`-elementen.

❖ Secties

Het `section`-element wordt gebruikt om verschillende secties aan te duiden van een geheel. Een sectie is een onderverdeling die op zichzelf onvolledig is, bijv. een hoofdstuk in een boek of de eerste pagina van een blogpost met meerdere pagina's. Dit element is soms moeilijk exact te definiëren en komt eigenlijk het dichtst tegen het `div`-element. Het verschil met een `article` is dat dat wel op zichzelf zou kunnen bestaan (zie Artikel). Het verschil met `aside` is dat een `aside` kan worden weggelaten, het is 'extra' (zie Aside). Een `section` kan andere `sections` en `articles` bevatten, die op hun beurt weer `deelsections` of `-articles` bevatten, ...

❖ Artikel

Het `article`-element geeft een groep elementen aan met samenhangende inhoud binnen de huidige groep. Een leidraad om te kiezen voor een artikel is de vraag of de inhoud op zichzelf gelezen of gedeeld kan worden.

Soms wordt aangeraden het `article`-element vooral te gebruiken voor inhoud die van eventueel van buitenaf zou kunnen worden ingevoegd. In geval van een krantenartikel, een forumpost, een nieuwsitem, enz. is dit dus een goede keuze.

```
<section class="agenda">
  <article class="eventement">...</article>
  <article class="eventement">...</article>
</section>
<section class="terugblik">
  <article class="eventement verslag">...</article>
  <article class="eventement verslag">
    <section class="fotoboek">...</section>
    <section class="interviews">
      <article>...</article>
      <article>
        <p>...</p>
        <section class="comments">...</section>
      </article>
    </section>
  </article>
</section>
```

❖ Hoofding

Het `header`-element is veel gemakkelijker betekenisvol te plaatsen. Het wordt gebruikt voor hulpmiddelen met een inleidend of verwijzend karakter (niet te verwarren met het `<head>`-element). Bij uitgebreide webpagina's zal het `header`-element dus meerdere malen aanwezig zijn. Het eerste kinderelement binnen de `header` is meestal een kopelement (`h1`, `h2`, `h3`, ...). Merk op dat `header`-elementen niet altijd bovenaan staan.

```
<header>
  <h1>Pagina titel</h1>
  <h2>Subtitel</h2>
</header>
```


❖ Voettekst

Het `footer`-element geeft voettekst aan, met meta- of extra detailinformatie. Ook dit element kan per pagina dus meerdere keren voorkomen. Binnen de footer van een webpagina wordt vaak de copyrightinformatie, links naar gerelateerde documenten en contactgegevens opgenomen. Bij artikels wordt in de footer-tag vaak de auteur, de aanmaakdatum, enzovoort opgenomen.

```
<article>
  <h2>Artikel</h2>
  <footer>
    Auteur: <address><a href="mailto:naam@mail.yy">Naam</a></address>
  </footer>
  <p>De eigenlijke inhoud.</p>
</article>
```

❖ Navigatieblok

De `<nav>...</nav>`-tags omvatten het hoofdmenu met navigatielinks van de webpagina, maar kan verderop ook gebruikt worden voor submenu's, inhoudstafels, ... Het `nav`-element kan dus eveneens meerdere keren op een webpagina voorkomen.

```
<nav>
  <ul>
    <li><a href="#">Hyperlink 1</a></li>
    <li><a href="#">Hyperlink 2</a></li>
    <li><a href="#">Hyperlink 3</a></li>
  </ul>
</nav>
```

❖ Randinformatie

Het `aside`-element kan gebruikt worden om een paginafragment aan te duiden dat zich in de marge van de hoofdpagina bevindt. Het betreft dan meestal een extra kolom met bijkomende informatie die niet rechtstreeks gerelateerd is aan de eigenlijke inhoud. Een voorbeeld is een banner, reclame, een figuur, een wist-je-datje, enzovoort.

```
<aside>
  <h2>Ook dat nog...</h2>
  <p>Wist je dat aside niet hetzelfde is als offside?</p>
</aside>
```

Een mogelijke indeling van een webpagina met bovenstaande semantische elementen kan er dus als volgt uitzien. Dit voorbeeld is slechts één van de vele mogelijkheden met de nieuwe elementen.

```
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="utf-8">
  <title>Structuurelementen</title>
</head>
<body>
  <header>
    <h1>Paginatitel</h1>
    <p>Tekst</p>
  </header>

  <nav>
    <ul>
      <li><a href="link1.html">Link 1</a></li>
      <li><a href="link2.html">Link 2</a></li>
      <li><a href="link3.html">Link 3</a></li>
    </ul>
  </nav>
```

```
<section>
  <header>
    <h2>Titel </h2>
  </header>
  <article>
    <header>
      <h3>Artikel titel </h3>
    </header>
    <p>Artikel inhoud 1</p>
    <footer><p>Artikel voetnoot<p></footer>
  </article>
  <article>
    <p>Artikel inhoud 2</p>
  </article>
  <article>
    <p>Artikel inhoud 3</p>
  </article>
</section>

<aside>
  <p>Bijkomende inhoud</p>
</aside>

<footer>
  <p>Pagina voetnoot</p>
</footer>
</body>
</html>
```

2.3 Inline-elementen

Zoals de naam het al aangeeft worden **inline-elementen** allemaal binnen dezelfde regel geplaatst. Ze zijn niet gescheiden door een overgang naar een nieuwe regel, maar staan naast elkaar.

❖ Line break

Indien je inline elementen te midden van een paragraaf toch wil scheiden, kan je dit doen door een line break te plaatsen. De line break dient om doelbewust een geregeleinde te creëren zonder de bij de <p>-tag horende blanco regel. Dit element kan geen inhoud bevatten, dus in de HTML-syntax is de sluittag niet nodig.

```
<br>
```

❖ Word break

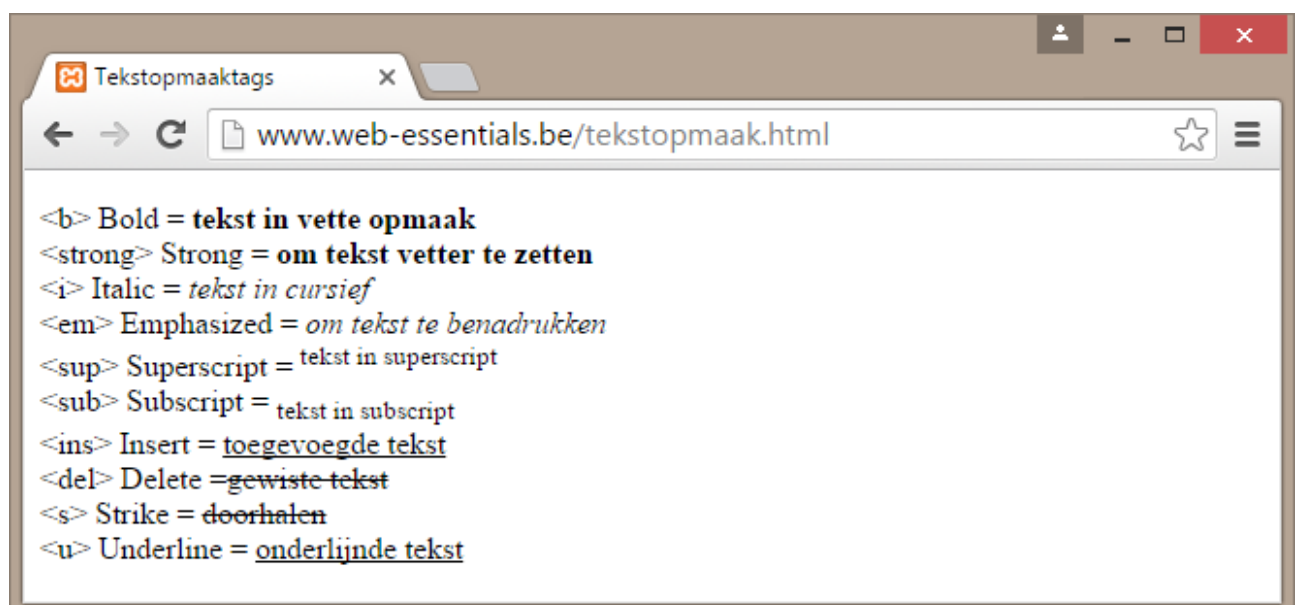
Met het wbr-element (word break) kan je mogelijke plaatsen aangeven om een woord af te breken indien het woord te lang wordt om op de regel te passen. Dit geeft echter geen koppeltokens. Gebruik daarvoor het bijhorende Unicode teken U+00AD 'soft hyphen'.

```
<p>Als je hippopotam<wbr />monstro<wbr />sesquipedalia<wbr />fobie hebt,  
dan heb je angst voor lange woorden. </p>
```

❖ Teksttags

Alle teksttags zijn inline-elementen en geven de auteur zeggenschap over de betekenis van die tekst. Deze betekenis kan soms erg subtiel zijn. De tags hebben ook een standaardopmaak, te zien in de onderstaande figuur.

...	Belangrijk, urgent
...	Beklemtoond
^{...}	Superscript
_{...}	Subscript
<ins>...</ins>	Ingevoegd
...	Gewist
<s>...</s>	Niet langer relevant
...	Uitgelicht, maar zonder extra betekenis
<i>...</i>	Andere stemming
<u>...</u>	Met een onuitgesproken niet-tekstuele annotatie



Het ``- en ``-element resulteren in dezelfde vette opmaak. Het geniet echter altijd de voorkeur om betekenis te geven aan de inhoud, dus indien je iets belangrijk wil maken is `` de betere keuze. Hetzelfde geldt voor `<i>` en `` die beiden een cursieve opmaak geven, maar waar de tekst een andere betekenis krijgt door de klemtoon op een deel te leggen, gebruik je het ``-element. Waar de tekst echter een andere stemming weergeeft (bijv. een droom) dan de omgevende tekst, is het `<i>`-element gepast.

De `<big>` en `<u>`-elementen zijn afgekeurd in HTML5. Het is trouwens af te raden om onderlijnde tekst te gebruiken aangezien dit verwarring geeft met hyperlinks die standaard onderlijnd zijn. Als je louter een visueel effect wilt bekomen gebruik je het element met de meest passende betekenis of het ``-element samen met de bijhorende CSS-code.

Om revisies aan te geven kunnen de elementen `<ins>` en `` dienen. Gewiste tekst wordt aangeduid met `` (deleted) en toegevoegde tekst met `<ins>` (inserted). Enkel als je tekst wil aangeven die niet langer geldig of relevant is, is het `<s>`-element aangewezen.

❖ Logische elementen

Met deze elementen wordt een logische betekenis gegeven aan flarden tekst. De verdere uitwerking ervan hangt af van de gebruikte browser.

De betekenis kan computeruitvoer gebaseerd zijn:

<code><code>...</code></code>	om computercode weer te geven
<code><kbd>...</kbd></code>	om gebruikersinvoer weer te geven
<code><samp>...</samp></code>	om output van een programma weer te geven
<code><var>...</var></code>	om variabelen weer te geven

Of kan verwijzen naar citaten, definities en referenties:

<code><abbr>...</abbr></code>	verkorte weergave (afkorting, acroniem, ...)
<code><bdi>...</bdi></code>	isoleert een stukje tekst voor de tekstrichting (bidirectional isolation)
<code><bdo>...</bdo></code>	richting van de tekst (bidirectional override)
<code><cite>...</cite></code>	voor citeren van de titel van een werk (lied, blogpost, programma, ...)
<code><dfn>...</dfn></code>	voor definities.
<code><small>...</small></code>	Korte nevenbemerkingen
<code><q>...</q></code>	Quotes, te midden van andere tekst



Zoals in bovenstaande figuur blijkt resulteren de elementen `<code>...</code>`, `<kbd>...</kbd>` en `<samp>...</samp>` in een lettertype met vaste letterbreedte (niet-proportioneel).

❖ Span

Indien je een stukje tekst wil opmaken met CSS zonder specifieke betekenis en zonder dat er al enige opmaak wordt meegegeven, kan je dit doen door gebruik te maken van het `span`-element.

```
<p> Met span kan je een bepaald <span class="...">stukje tekst</span> een andere opmaak meegeven </p>
```

2.4 Speciale tekens

Voor het gebruik van speciale symbolen zijn er codes ontworpen, de **entiteitsnamen**. Een entiteitsnaam begint met een ampersand, bevat een afgekorte benaming en eindigt op een puntkomma. In plaats van de naam kan ook de decimale of hexadecimale ASCII- of Unicodenummer gebruikt worden.

<	<	<	<	kleiner dan
>	>	>	>	groter dan
&	&	&	&	ampersand
een spatie	 	 	 	non breaking space
"	"	"	"	dubbel aanhalingsteken

Er zijn veel meer entiteiten, een volledige lijst van de mogelijkheden kan je terugvinden op <https://html.spec.whatwg.org/multipage/syntax.html#named-character-references>.

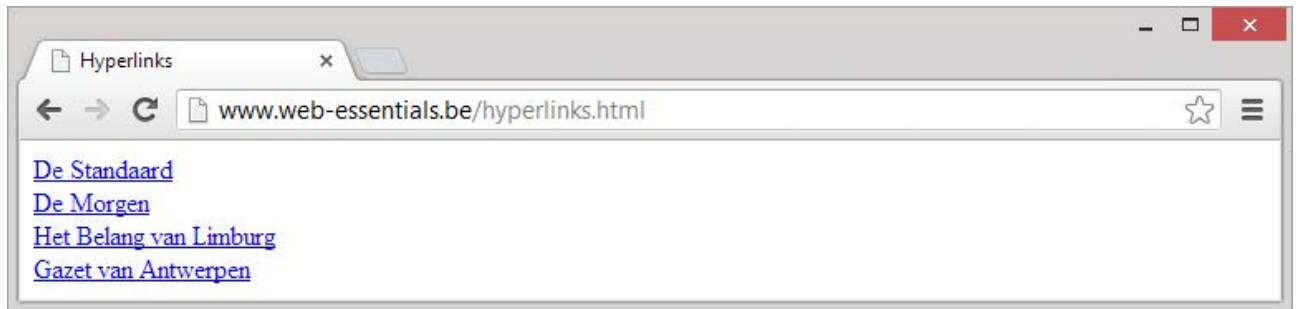
Voor andere karakters dan diegenen in de tabel hierboven is het makkelijker om de html-code met karakterset UTF-8 te maken en de karakters gewoon letterlijk in te geven, eventueel met hulp van een virtueel toetsenbord, kopiëren-plakken of <http://www.unicode.org/charts/charindex.html>

3 Hyperlinks

3.1 Verbindingen leggen

Om een webpagina te laten verwijzen naar een ander hypertext item gebruik je het **anchor** element `<a>...`. Het meest gebruikte attribuut van `<a>` is `href`, de **hypertext reference**.

```
<a href="http://www.destandaard.be">De Standaard</a><br>
<a href="http://www.demorgen.be">De Morgen</a><br>
<a href="http://www.hbvl.be">Het Belang van Limburg</a><br>
<a href="http://www.gva.be">Gazet van Antwerpen</a><br>
```



De andere attributen van `<a>` kunnen alleen samen met het `href`-attribuut aanwezig zijn.

HTML	waarde
<code>href</code>	De hyperlink die geopend wordt.
<code>hreflang</code>	De taal van de hyperlinkreferentie.
<code>media</code>	Geeft het medium/apparaat aan (vb. screen, pda, ...).
<code>rel</code>	De relatie tussen het document en de doel-url: alternate author bookmark help license next nofollow norereferrer prefetch prev search tag.
<code>target</code>	Doel waar de link geopend wordt: _blank _parent _self _top.
<code>type</code>	MIME-type van de doel-url.

❖ Opmaak van links

De opmaak van hyperlinks dient met pseudoklassen gedefinieerd te worden in CSS. Standaard zijn links altijd onderlijnd. De kleur geeft de status aan, niet-bezochte links zijn blauw, bezochte links paars en actieve links rood.

```
a {text-decoration: none;}
a: link {color: #0000ff;}
a: visited {color: #000099;}
a: hover {color: #ffffff; background-color: #0000ff;}
a: active {color: #000099;}
```

3.2 Absolute en relatieve koppelingen

Een **absolute koppeling** verwijst naar een bepaald document op een webserver en gebruikt hiervoor een URL-adres. Als van het bestand de naam wordt gewijzigd, als het naar een andere directory of op een andere server wordt geplaatst is de betovering verbroken en volgt er bij het aanklikken een foutmelding.

```
<a href="http://www.voorbeeld.be/index.html">...</a>
```

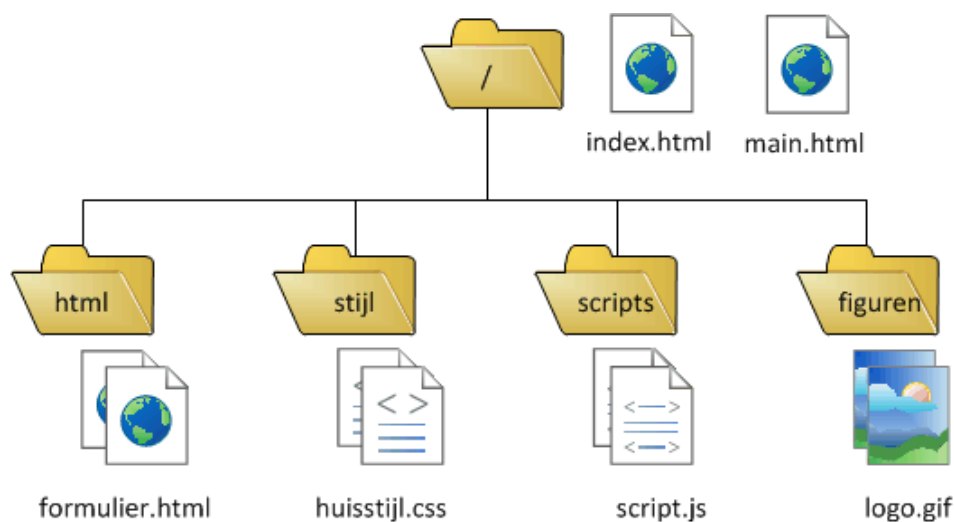
Een **documentrelatieve koppeling** verwijst naar een ander bestand, maar doet dit relatief ten opzichte van het bestand van waaruit de link wordt aangeklikt. Het voordeel hiervan is dat de directories in hun geheel, met al de onderliggende directories naar een andere computer overgebracht kunnen worden waarbij de structuur en de koppeling behouden blijft.

Indien vanuit het bestand index.html dient gelinkt te worden naar het bestand main.html dat zich in dezelfde directory bevindt, dan geeft dit volgende syntax.

```
<a href="main.html">...</a>
```

Indien vanuit het bestand index.html dient gelinkt te worden naar formulier.html dat zich in de directory html bevindt, dan dien je de foldernaam mee op te nemen. Let erop dat voorwaartse slashes gebruikt worden om foldernamen aan te geven. Moeten er meerdere niveaus gedaald worden, dan worden alle foldernamen gescheiden door een slash (dus **geen backslash**).

```
<a href="html/formulier.html"> ...</a>
```



Indien er in de folderhiërarchie dient gestegen te worden dan maak je gebruik van ../. Zo kan je van formulier.html linken naar index.html met de volgende syntax.

```
<a href="../../index.html">...</a>
```

Het benaderen van de figuur logo.gif in het huisstijl.css bestand in de stijlmap ziet er dus zo uit.

```
div {
  background-image: url(../figuren/logo.gif);
}
```

Een andere methode om relatieve links te leggen is met een **site-rootrelatieve koppeling**. Hierbij wordt de rootdirectory van de site aangeduid door een slash, en vertrek je altijd van deze locatie.

Enkele voorbeelden van site-rootrelatieve koppelingen:

```
<a href="/html/formulier.html">...</a>

<a href="/index.html">...</a>
<link rel="stylesheet" href="/stijl/huisstijl.css">
```

De site-rootrelatieve links werken enkel goed indien de bestanden reeds op een server geplaatst zijn en kunnen dus slecht getest worden wanneer ze op de lokale harde schijf staan.

3.3 Interne koppelingen

Een **interne koppeling** wordt gebruikt om een sprong te maken binnen een document. Hiermee worden dus ankers bedoeld die je naar een andere positie op dezelfde of een andere pagina brengen.

Een interne koppeling bestaat uit twee delen, een anker met naam `...` en een hyperlink naar die naam toe `...`.

```
<h2>Inhoudsopgave</h2>
<ul>
  <li><a href="#deel 1">deel 1</a></li>
  <li><a href="#deel 2">deel 2</a></li>
</ul>
<article>
  <h3><a id="deel 1">Deel 1: Inleiding</a></h3>
</article>
<article>
  <h3><a id="deel 2">Deel 2: Structuur</a></h3>
</article>
```

De ankerkoppeling met de hypertext reference wordt door de browser onderstreept weergegeven, die met het `id`-attribuut wordt echter niet onderstreept. Beiden worden afgesloten met ``.

In de oudere HTML-standaarden wordt gebruik gemaakt van het `name`-attribuut, tegenwoordig geniet het `id`-attribuut de voorkeur. In HTML5 hoeft er zelfs geen anker meer aanwezig te zijn en kan je het vermelde `id` ook aan een ander element toekennen. Onderstaande code werkt dus hetzelfde als de bovenstaande.

```
<h2>Inhoudsopgave</h2>
<ul>
  <li><a href="#deel 1">deel 1</a></li>
  <li><a href="#deel 2">deel 2</a></li>
</ul>
<article id="deel 1">
  <h3>Deel 1: Inleiding</h3>
</article>
<article id="deel 2">
  <h3>Deel 2: Structuur</h3>
</article>
```

Er kan eveneens vanuit een ander bestand gelinkt worden naar een interne link van een bestand.

```
<a href="index.html#deel 1">deel 1</a><br>
```

Een veelgebruikte toepassing is de top-link in webpagina's met veel inhoud waarbij een verticale scrollbar nodig is om alle inhoud te kunnen zien. Hierbij wordt er bovenaan de pagina een `id="top"` geplaatst en onderaan een link naar die toplocatie.

```
<h1><a id="top">Hoofdstuk 1</a></h1>
...
<a href="#top">Top</a>
```

Het plaatsen van het topanker is strikt genomen niet noodzakelijk omdat `href="#"` sowieso naar het begin van de pagina verwijst.

```
<h1>Hoofdstuk 1</h1>
...
<a href="#">Top</a>
```


3.4 Andere soorten koppelingen

❖ Koppeling naar een e-mailadres

Hiervoor wordt het protocol `mailto` gebruikt met `naam@systeem.be` om op te geven wie de e-mail dient te ontvangen.

```
<a href="mailto:naam@systeem.be">...</a>
```

De `mailto`-opdracht kan uitgebreid worden met een “cc” (carbon copy), een “bcc” (blind carbon copy) en/of een “subject”. Dit wordt ingegeven door achter de geadresseerde een vraagteken in te geven. Daarna komt de “cc”, de “bcc” en/of de “subject” gescheiden door een ampersand.

```
<a href="mailto:naam@systeem.be?cc=iemandanders@bedrijf.be&bcc=nogiemand@bedrijf2.be&subject=Dringende%20boodschap!">...</a>
```

Merk op dat `&` geëncodeerd is als entiteit `&` omdat `&` op zich niet mag voorkomen in HTML. Verder is de spatie in het onderwerp geëncodeerd als `%20`, omdat een spatie niet mag voorkomen in een URL.

❖ Bestanden downloaden

Voor veel bestanden is het niet nodig een server met een apart protocol op te zetten. Door middel van een normale hyperlink kan je zip-bestanden, mp3-muziek, pdf-bestanden, videofragmenten (*.wmv, *.mov), Word-documenten, enzovoort downloaden.

Als de browser het bestandstype zelf ondersteunt kan het bestand geopend worden in een aparte webpagina of tab. In het andere geval krijg je niet altijd duidelijk de keuze het bestand op te slaan. Met pagina opslaan (Ctrl+S) lukt dit meestal wel. Rechtsklikken op de hyperlink biedt vaak ook deze mogelijkheid.

```
<a href="figuren.zip">download de figuren</a>
<a href="cursus.pdf">download de cursus</a>
```

4 Cascading Style Sheets

In de eerste plaats is HTML ontwikkeld om de structuur van een pagina te beschrijven en niet zozeer voor de opmaak. Hierdoor heeft HTML aardig wat beperkingen. Om nu ook een mechanisme te hebben om de opmaak en layout van een pagina nauwkeurig te controleren, heeft W3C het concept van stylesheets bedacht. Dit zijn beschrijvingen hoe de stijl van een pagina eruit moet zien, die losgekoppeld zijn van de structuur van een pagina.

Het W3C heeft in 1996 de standaard van CSS level 1 (CSS1) beschreven. CSS staat voor **Cascading Style Sheets**, wat betekent dat er verschillende niveaus van stylesheets zijn: lokaal, globaal en extern. Bij conflicten heeft het lokale niveau voorrang over het globale en het globale over het externe niveau. Aangezien deze standaard al tamelijk oud is wordt ze door haast alle hedendaagse browsers ondersteund (vanaf Internet Explorer 3.0 en van Netscape Communicator 4.0). In 1998 werd CSS1 uitgebreid naar **CSS2** (ondersteund vanaf Internet Explorer 5.0 en van Netscape Communicator 6.0). De huidige versie is **CSS3**, dat is opgebouwd uit verschillende modules (bijv. CSS3 Animation, CSS3 Borders, ...).

Met stylesheets kan worden gespecificeerd hoe bepaalde elementen op het scherm eruit zullen zien, bijvoorbeeld welke kleur een tekst heeft, welk font gebruikt moet worden, hoe groot het font is, wat de spatiëring is, welke achtergrond, Een aantal van zulke eigenschappen bij elkaar wordt een stijl genoemd.

De grote voordelen van stylesheets zijn dus:

- De mogelijkheid om inhoud en stijl te scheiden;
- Het wegwerken van de bestaande tekortkomingen in HTML, XHTML en HTML5;
- De mogelijkheid om alle “deprecated” elementen te vervangen en zo HTML5-gevalideerde pagina's te maken;
- Een stijl kan van toepassing zijn op verschillende elementen, bijvoorbeeld op `<p>`, ``, `<body>` of `<h2>`.

4.1 De syntax van CSS

Een CSS-regel bestaat uit twee delen; een **selector** die gevolgd wordt door de **stijldeclaratie**. De selector kan bestaan uit een HTML-element, een klasse, een id en zelfs uit meerdere selectors gescheiden door een komma. De stijldeclaratie komt tussen accolades en indien er meerdere zijn worden ze gescheiden door een puntkomma.

```
p {
    font-size: 15pt;
    line-height: 20pt;
    margin-left: 1in;
    margin-right: 1in;
    text-indent: .5in;
}

h1, h2 {
    font-family: "Times New Roman", serif;
}
```

❖ Verkorte schrijfwijze CSS

Een opeenvolging van bij elkaar horende stijlen kan ook met een verkorte weergave geschreven worden. Hierbij is bij sommige stijldeclaraties (zoals *font*) de volgorde echter wel belangrijk, bij anderen heeft de volgorde geen invloed.

Voor de fontdeclaratie moeten *weight* en *style* bijvoorbeeld eerst gespecificeerd worden. Lettertypen binnen een *font-family* die bestaan uit meer dan één woord moeten tussen

aanhalingstekens geschreven worden. Bij de borderdeclaratie speelt de volgorde dan weer geen rol.

```
p {
  border-style: solid;
  border-width: 2px 10px 4px 20px;
  border-color: red;

  font-family: Times, serif;
  font-size: 12pt;
  line-height: 20pt;
  font-weight: bold;
  font-style: italic;
}
```

Of met verkorte schrijfwijze:

```
p {
  border: 2px dashed red;
  font: bold italic 12pt/20pt Times, serif;
}
```

❖ Commentaar

Commentaar kan eveneens toegevoegd worden, maar moet dan omgeven zijn door `/*...*/`.

```
/* commentaar */
```

❖ Important

Om ervoor te zorgen dat er geen rekening gehouden wordt met de cascadevolgorde kan een stijldeclaratie als belangrijk aangeduid worden.

```
p {
  font-size: 12pt !important;
}
```

4.2 Toepassing van Cascading Style Sheets

❖ Lokale stijl: binnen elk HTML-element

Bij het aanroepen van een HTML-element, dus binnen de starttag, kan een stijldeclaratie geplaatst worden. In deze `style="..."` code worden een aantal eigenschappen bepaald. Dit noemt men een **lokale** of **inline stijl**. De opgenomen stijlbeschrijving geldt enkel voor dat HTML-element. Het gebruik van een lokale stijldeclaratie kan in combinatie met attributen gebeuren.

```
<p style="color: red; font-family: 'New Century Schoolbook', serif; font-size: large;"> This paragraph is styled in red with the New Century Schoolbook font, if available. </p>
```

❖ Globale stijl: in de HTML-pagina

Een globale stijl of **blokstijl** wordt opgenomen in het `<head>`- gedeelte van een pagina door middel van `<style>`- code waarin de opmaak van de HTML-elementen bepaald wordt. De `<style>`-tag heeft een attribuut type die in html5 als default waarde `"text/css"` heeft.

Telkens dit element in de `<body>` gebruikt wordt zullen deze eigenschappen overgenomen worden (**globaal**). Deze eigenschappen gelden enkel binnen het HTML-document waarin de blokstijl is opgenomen.

```
<style>
body {
    margin: 10px;
    background-color: #999999;
}

p {
    font: 10pt/14pt Verdana, Arial, Geneva, sans-Serif;
    margin: 1em .5em;
    text-indent: 15px;
}
</style>
```

❖ Gelinkte stijl: apart stijlblad

Een **extern** stijlbestand bevat vele stijldefinities en heeft meestal .css als extensie. Een stijlbestand heeft dezelfde opmaak als een globaal stijlblad in de <head>.

Het linken van dit CSS-bestand aan meerdere bestaande HTML-bestanden gebeurt door het opnemen van het <link>-element in de <head> van die HTML-bestanden. Binnen <link> moet je verschillende attributen opnemen. Het rel - attribuut bepaalt de relatie met het document en de gebruikte waarde is "stylesheet". Het href - attribuut bepaalt welk stijlblad dient gekoppeld te worden. Het juiste type kan worden aangegeven door type="text/css", maar dat is niet langer nodig in HTML5 aangezien de relatie al aangeduid wordt door het rel -attribuut.

```
<link rel="stylesheet" href="stijlbestand.css"/>
```

Met rel="alternate stylesheet" kan je alternatieve stijlbestanden laden.

Een alternatief voor het <link>-element is het gebruik van @import om een extern stijlbestand te koppelen. Een voordeel is dat zo meerdere stijlbestanden eenvoudig kunnen geïmporteerd worden. Een nadeel is dat @import niet ondersteund wordt door sommige oudere browsers. De code om hetzelfde resultaat te bekomen als het hierboven vermeld voorbeeld wordt dan:

```
<style type="text/css">
    @import url(stijlbestand.css);
</style>
```

Een handige toepassing van @import is om het gezamenlijk te gebruiken met het <link>-element. Op deze wijze kunnen externe stijlbestanden in elkaar opgeroepen worden.

In de HTML-code is dan enkel een link te zien:

```
<link rel="stylesheet" href="support.css"/>
```

In het opgeroepen extern stijlbestand worden bovenaan één of meerdere andere stijlbestanden geïmporteerd.

```
@import url(huisstijl.css); /* invoegen van de huisstijl */
/* hierna komen alle stijlen voor de supportpagina */
```

4.3 De waterval en overerving

In de term "cascading style sheets" verwijst de term "cascade" naar het watervalstelsel dat geldt bij het toekennen van stijlen. Zoals hierboven vermeld kunnen de stijlen onderverdeeld worden in elementspecifieke stijlen (lokaal), paginaspecifieke stijlen (globaal) en sitespecifieke stijlen (gelinkt). Het principe van een waterval is dat het water van allerlei plaatsen kan toestromen maar steeds naar beneden stroomt. Ditzelfde effect wordt bekomen door de verschillende niveau's van stijlen te laten samenvloeien met als eindpunt het HTML-element. Het hoogste niveau vanwaar stijlen kunnen komen is het gelinkte stijlbestand, daarna komt de paginaspecifieke stijl en

uiteindelijke de elementspecifieke stijl. Als telkens andere opmaken meegegeven worden, cumuleren die uiteindelijk tot een samengestelde stijl.

Als bijvoorbeeld voor een paragraaf in het extern CSS-bestand *font-family: Arial* is meegegeven, en in de <head> staat bij de paragraafopmaak *font-size: 12pt* en lokaal bij een bepaalde paragraaf in de HTML-code staat `style="color: blue"` dan zal de paragraaf uiteindelijk de drie kenmerken vertonen, dus in Arial, met een grootte van 12pt en in het blauw.

Stel dat in het extern CSS-bestand *font-family: Arial* niet ingesteld was voor het <p>- element, maar wel voor het <body>- element, dan zal het resultaat niet verschillen. Elke paragraaf is immers een **child** van de **parent** body, waardoor deze kenmerken worden overgeërfd. Van deze eigenschap kan handig gebruik gemaakt worden om bepaalde eigenschappen voor meerdere elementen in te stellen. Dit is zeker het geval bij geneste elementen.

4.4 Classes

Stylesheets zijn een zeer krachtig hulpmiddel om een huisstijl af te dwingen op een website. Immers, er kan vanuit elke pagina een link worden gelegd naar een uitgebreide externe stylesheet, die de huisstijl exact definieert. Naast het afdwingen van een stijl scheelt het ook veel programmeerwerk, want het werk wordt zo op één plaats gedaan. In de webdocumenten zelf kan dan alle aandacht uitgaan naar de structuur in plaats van de opmaak.

Om van een dergelijke tweedeling nog meer profijt te hebben, zijn er bij stylesheets ook zogenoemde klassen geïntroduceerd. Dit houdt in dat er een reeks van eigenschappen worden ondergebracht onder één noemer in een klasse in plaats van dat ze worden toegekend aan een bepaalde HTML-tag, zoals je in de voorgaande voorbeelden hebt gezien. Indien stijlen per element dienen beschreven te worden, is het immers noodzakelijk om stijldeclaraties meerdere malen in een stijlblad op te nemen. Door klassen te gebruiken voorkom je deze herhaling.

❖ Klassen (classes)

Binnen een klasse kan je een stijl beschrijven, die je daarna op willekeurige en meerdere elementen kan toepassen. De klasse komt binnen de stijlblok en wordt aan het element gekoppeld door het `class`-attribuut te beschrijven.

De algemene syntax van een klasseregel start met de klasseselector die start met een punt. Na de klasseselector komt de stijldeclaratie

```
. naam {
    ei genschap1: waarde;
    ei genschap2: waarde;
    ei genschap3: waarde;
}
```

Een uitgewerkt voorbeeld zal de werking duidelijk maken.

Het HTML-bestand:

```
<head>
  <link rel="stylesheet" href="stijl.css" />
</head>
<body>
  <p class="text">The class of text. </p>
  <p class="red">This is red color with size 13. </p>
  <p class="text">. text can be used again! </p>
  <p class="red">. red can be used again! </p>
</body>
```

Het stijlbestand "stijl.css":

```
.text {
    font-family: "Verdana", "Arial", "Times New Roman";
    font-size: 10pt;
}

.red {
    color: #FF0000;
    font: bold 13pt;
}
```

Je kan bijvoorbeeld een klasse met de naam 'huisstijl' aan maken.

```
.huisstijl {
    color: #00FF00;
    letter-spacing: 0.1em;
    word-spacing: 0.4em;
    text-indent: 3em;
    text-align: center;
    background-image: url (back.gif);
    background-repeat: repeat-y;
    background-attachment: fixed;
    background-position: 100% 100%;
}
```

Als een dergelijke klasse is gedefinieerd kan je er binnen HTML-tags naar refereren door de `class=""` vermelding. Indien je de huisstijl binnen het hele document wil laten gelden, neem je een verwijzing op in de `<body>`-tag.

```
<body class="huisstijl">
```

❖ Afhankelijke klassen (fixed classes)

Klassen kunnen ook gekoppeld worden aan een bepaald HTML-element. Op deze wijze kan je stijlen definiëren die enkel voor dat element gelden. De syntax van een afhankelijke klasseregels is identiek aan de gewone klasse. Het enige verschil is dat dit gekoppeld wordt aan een bepaald HTML-element.

```
element.naam {
    eigenschap1: waarde;
    eigenschap2: waarde;
    eigenschap3: waarde;
}
```

Voorbeeld:

CSS:

```
p.red {
    color: #FF0000;
    font-size: 13pt;
    font-weight: bold;
}
```

HTML:

```
<body>
  <p class="red">Enkel paragrafen kunnen gebruik maken van p.red.</p>
</body>
```

❖ Pseudoklassen

Pseudoklassen definiëren de opmaak van het element. In CSS1 werden deze pseudoklassen enkel toegepast op het <a>-element. Men spreekt van pseudo-classes omdat het attribuut `class` niet opgenomen wordt bij het openen van het element. De browser bepaalt of de link al dan niet bezocht is, actief staat of aangewezen wordt en zal dan de juiste opmaak toepassen.

<code>a: link</code>	<code>{ color: #0000FF; text-decoration: none; }</code>
<code>a: visited</code>	<code>{ color: #0000AA; text-decoration: underline; }</code>
<code>a: hover</code>	<code>{ color: #0000FF; text-decoration: underline; }</code>
<code>a: active</code>	<code>{ color: #0000AA; text-decoration: underline; }</code>

Hier volgt een overzicht van enkele pseudoklassen. De *focus*-pseudoklasse zal een stijl toepassen wanneer het element de focus heeft. Dit is vooral handig in formulieren. *:first-child* geeft het eerste kind van een element de beschreven stijl. De pseudoklasse *lang* laat de auteur toe een bepaalde taal te koppelen aan een element.

CSS	selectie	voorbeeld
<code>:link</code>	een niet-bezochte koppeling	<code>a:link</code>
<code>:visited</code>	een bezochte koppeling	<code>a:visited</code>
<code>:hover</code>	het element waar de muiscursor boven staat	<code>img:hover</code>
<code>:active</code>	het actieve element (meestal de koppeling waarop geklikt is)	<code>a:active</code>
<code>:focus</code>	het element met de focus	<code>input:focus</code>
<code>:first-child</code>	het eerste kindelement van de parent	<code>p:first-child</code>
<code>:lang()</code>	het element met de juiste waarde in het lang-attribuut	<code>p:lang(nl)</code>
<code>:empty</code>	Een leeg element.	<code>p:empty</code>
<code>:not()</code>	Een element dat niet aan de voorwaarde voldoet.	<code>input:not([type=radio])</code>
<code>:enabled</code>	Een invoerelement dat enabled is.	<code>input:enabled</code>
<code>:disabled</code>	Een invoerelement dat disabled is.	<code>input:disabled</code>
<code>:checked</code>	Een invoerelement (radio, checkbox) dat checked is.	<code>input:checked</code>
<code>:first-of-type</code>	Het eerste element van dat type.	<code>p:first-of-type</code>
<code>:last-of-type</code>	Het laatste element van dat type.	<code>p:last-of-type</code>
<code>:only-of-type</code>	Het enige element van dat type.	<code>p:only-of-type</code>
<code>:last-child</code>	Het laatste kind van dat element.	<code>div:last-child</code>
<code>:only-child</code>	Het enige kind van dat element.	<code>div:only-child</code>
<code>:nth-child()</code>	Het zoveelste kind van dat element.	<code>div:nth-child(2n)</code>
<code>:nth-last-child()</code>	Het zoveelste kind van dat element, maar teruggellend.	<code>div:nth-last-child(2)</code>
<code>:nth-of-type()</code>	Het zoveelste element van dat type.	<code>div:nth-of-type(2n)</code>
<code>:nth-last-of-type()</code>	Het zoveelste element van dat type, maar teruggellend.	<code>div:nth-last-of-type(2)</code>

❖ Pseudoklassen II: Fixed Pseudo-classes

Pseudoklassen van type II bekom je door afhankelijke klassen te definiëren voor pseudoklassen type I.

CSS:

```
a.bold:link {color: #FFFF00; text-decoration: underline; font-weight: bold;}
a.bold:visited {color: #FFFF00; text-decoration: none; font-weight: bold;}
a.bold:hover {color: #FFFF00; text-decoration: none; font-weight: normal;}
```

HTML:

```
<p><a class="bold" href="http://www.x.net/x.html">This is with a 'bold'
class</a></p>
<p><a class="bold" href="http://www.x.net/y.html"> The visited link with
'bold' class</a></p>
```

❖ Pseudo-elementen

Binnen CSS3 zijn ook pseudo-elementen gedefinieerd:

CSS	selectie	voorbeeld
::first-letter	De eerste letter van het element.	p::first-letter
::first-line	De eerste lijn van een paragraaf.	p::first-line
::before	Plaatst inhoud voor het element.	p::before
::after	Plaatst inhoud achter het element.	p::after

```
p::first-letter {font-size: 1.5em;}
p::first-line {font-size: 1.2em;}
```

```
p::before {content: url(figuur.gif);}
p::after {content: "bijkomende tekst"}
```

De website <http://www.quirksmode.org> is een goede site om te controleren welke pseudoklassen en -elementen door welke browser ondersteund worden.

4.5 ID's

Indien een stijldeclaratie slechts voor één enkel element bedoeld is, is het niet logisch om hiervoor een klasse te creëren. In dit geval worden id's gebruikt om een unieke identificatie toe te kennen aan een element. De algemene syntax van een id-regel start met de klasseselector die start met een #. Na de klasseselector komt de stijldeclaratie

```
#naam {
    eigenschap1: waarde;
    eigenschap2: waarde;
    eigenschap3: waarde;
}
```

In het HTML-document dient het id-attribuut toegevoegd te worden zodat er een koppeling ontstaat tussen de stijldeclaratie en het unieke element.

```
<p id="naam">enkel deze paragraaf zal bovenvermelde eigenschappen
verkrijgen. </p>
```


4.6 Speciale selectors

❖ Asterisk

Indien een bepaalde opmaak voor alle elementen dient gebruikt te worden, wordt het snel onoverzichtelijk om al die stijlen te koppelen. In dit geval kan het handig zijn om met een universele selector te werken. Het meegeven van een asterisk werkt als “wildcard” en refereert dus naar alle elementen.

```
* {margin-left: 10px; }
```

❖ Gecombineerde klassen

Een tweede speciale situatie is wanneer er meerdere klassen wensen toegepast worden. Dit kan eenvoudig bekomen worden door in de HTML-code alle klassenamen na elkaar te schrijven.

```
.geel {
  color: yellow;
}
.vet {
  font-weight: bold;
}
.cursief {
  font-style: italic;
}
```

```
<p class="geel cursief">Deze tekst is geel en cursief</p>
```

Een meer gevorderde methode is om de klassen reeds te combineren bij de opmaak van de stijldeclaraties. Zo kan ervoor gezorgd worden dat wanneer twee klassen gecombineerd worden gebruikt nog een extra eigenschap wordt meegegeven. Met onderstaande aanvulling wordt in dat geval een rode achtergrondkleur ingesteld.

```
p.geel.cursief {
  background-color: red;
}
```

❖ Contextuele selectors

Indien de HTML-elementen genest gebruikt worden is er een **parent-child** relatie tussen de elementen onderling. Deze verwantschap kan gebruikt worden om bepaalde elementen een opmaak mee te geven die afhankelijk is van het element waarin ze genest zijn, men spreekt dan van een contextuele selector.

```
div h1 strong {
  color: gray;
}
```

In bovenstaand voorbeeld krijgen dus enkel de **strong**-elementen die afstammen van een kop1 (**h1**) die in een divider (**div**) is opgenomen de vermelde stijl.

Indien enkel een pure ouder-kindrelatie gewenst is, kan dit gespecificeerd worden met het groter-dan-teken.

```
div>h1>strong {
  color: gray;
}
```

Een voorbeeld zal het verschil tussen beide verduidelijken.

```
<div>
  <h1>Dit is een <strong>heel</strong> belangrijke titel</h1>
</div>

<div>
  <h1><span>Dit is een <strong>heel</strong> belangrijke titel</span></h1>
</div>
```

Wanneer in dit voorbeeld het >-teken gebruikt wordt zal enkel in de eerste kop het woord “heel” een grijze kleur meekrijgen. Als spaties gebruikt worden zal dit in beide koppen gebeuren.

Aangezien deze contextuele selectors zeer strikt zijn kan het handig zijn om met een universele selector te werken.

```
#sectie1 * span {
  margin-left: 10px;
}
```

In bovenstaand voorbeeld krijgen alle span-elementen, die binnen om het eender welk ander element genest zijn binnen sectie1, de opgegeven stijl.

❖ Kinderen combineren

Naast de parent-child relatie kan er ook nog een **sibling** relatie zijn. In dit geval moeten de elementen als broers of zussen beschouwd worden van eenzelfde parent. Zo zijn bijvoorbeeld twee paragrafen die onder de body staan, beiden een kind van die body en dus siblings. Bij het opmaken van stijldeclaraties kan de volgorde van deze nazaten gebruikt worden door gebruik te maken van het plus-teken. Op deze wijze kan bijvoorbeeld ervoor gezorgd worden dat enkel de eerste paragraaf die volgt na een <h1>-kop zal inspringen.

```
h1 + p {
  text-indent: 10px;
}
```

❖ Attribuutselectors

Je kan ook een selectie maken op basis van een attribuut of attribuutwaarde van een element.

```
img[alt] {...} /* img-elementen met een alt-attribuut */
img[alt="figuur"] {...} /* img-elementen met alt="figuur" */
img[alt^="figuur"] {...} /* img-elementen met alt beginnend met "figuur" */
img[alt$="figuur"] {...} /* img-elementen met alt eindigend met "figuur" */
img[alt*="figuur"] {...} /* img-elementen met "figuur" in de alternative */
```

4.7 Eenheden

De eenheden kunnen opgesplitst worden in relatieve en absolute waarden.

De relatieve eenheden:

- **em** (de breedte van de letter “M” van het gebruikte lettertype)
- **ex** (de hoogte van de kleine letter “x” van het gebruikte lettertype)
- **px** (pixel, volgens de schermresolutie)

De absolute eenheden:

- **in** (inches; 1in=2.54cm)
- **cm** (centimeters; 1cm=10mm)

- **mm** (millimeters)
- **pt** (points; 1pt=1/72in), voornamelijk om lettergroottes aan te geven.
- **pc** (picas; 1pc=12pt), voornamelijk om lettergroottes aan te geven.

Indien een website gebouwd dient te worden die onafhankelijk van de gebruikte hardware en software dezelfde afmetingen moet hebben, dan valt de keuze op absolute eenheden.

Naast deze eenheden kan ook met percentages gewerkt worden. Hoe dit percentage geïnterpreteerd dient te worden hangt af van de gebruikte eigenschap en van de juiste positie van het HTML-element.

4.8 Mediatypes

Binnen CSS kunnen verschillende mediatypes opgegeven worden om te specificeren welke stijlbestanden voor welk medium bedoeld zijn. Indien het mediatype niet wordt meegegeven gelden de stijldeclaraties voor alle media. De namen van de mediatypes zijn niet hoofdlettergevoelig.

De opgegeven media-naam spreekt meestal voor zich:

- **all**: geschikt voor alle toestellen.
- **aural**: voor spraakbrowsers (text-to-speech synthesizers). Hierin kan gewerkt worden met verschillende stemmen en stemvariabelen.
- **braille**: voor weergave door braillesystemen.
- **embossed**: voor weergave door brailleprinters.
- **handheld**: voor mobiele apparaten met een klein, soms monochroom, scherm en een beperkte bandbreedte. Dus voor de browsers die voorzien zijn op een pda en een gsm.
- **print**: opmaak voor de printer of voor de afdrukvoorbeelden. Vooral de mogelijkheid om pagebreaks toe te voegen is hier interessant.
- **projection**: voor projectoren of voor het printen van transparanten.
- **screen**: voor het standaard kleurencomputerscherm.
- **tty**: voor teletypes en terminals met een niet-proportioneel lettertype. Aangezien deze apparaten op letters gebaseerd zijn is het niet mogelijk om de pixel als eenheid te gebruiken.
- **tv**: voor tv-toestellen en andere schermen met een lage resolutie, een beperkte scrollbaarheid en beschikbaar geluid.

De implementatie van het mediatype kan op verschillende manieren gebeuren. Ofwel wordt het als een attribuut opgenomen in het `link`-element.

```
<link rel="stylesheet" href="stijl.css" media="screen" />
<link rel="stylesheet" href="pda.css" media="handheld" />
```

Ofwel wordt het bepaald in het extern stijlbestand door `@media` te gebruiken.

```
@media print {
    body { background-color: #FFFFFF; }
}
@media screen {
    body { background-color: #333333; }
}
@media screen, print {
    body { font-size: 1pt; }
}
```

Bij het importeren van een extern stijlbestand in een ander stijlbestand.

```
@import url(pda.css) handheld; /* invoegen van de pda-stijl */
```

5 Opmaak

5.1 Kleuren

Als er niets wordt opgegeven is de standaardkleur die door de browsers wordt gebruikt wit voor de achtergrond en zwart voor de voorgrond. Om een goed ogende webpagina te maken zijn natuurlijk meerdere kleuren nodig. Kleurkeuze van tekst en achtergrond zijn van belang voor de typografische leesbaarheid van webdocumenten.

❖ Kleurnamen

In de specificaties van CSS en HTML zijn **17 kleurnamen** vastgelegd: **aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white en yellow.** Vanaf CSS 3 zijn er 147 kleurnamen.

```
body {
  background-color: silver;
  color: black;
}
```

❖ RGB-model

Aangezien het zeer moeilijk (quasi onmogelijk) is om met de 17 vastgelegde kleuren een mooie webpagina te maken kunnen kleuren zelf gemengd worden. Hierbij kan je zelf bepalen welke intensiteiten van rood, groen en blauw licht gemengd worden. Je kan hiervoor gebruik maken van het hexadecimaal stelsel (een zestientallig systeem dat van de cijfers 0 tot en met 9 en de letters A tot en met F gebruik maakt om de getallen 0 tot en met 15 binnen het zestiental aan te duiden). Elke kleur kan dan variëren van 00h tot ffh en de notatie die gebruikt wordt is **#rrggbb**. Indien de twee symbolen voor een kleur hetzelfde zijn, is ook een verkorte schrijfwijze mogelijk, zo verwijzen bijvoorbeeld #AA33CC en #A3C naar dezelfde kleurwaarde. Op deze wijze kunnen alle benodigde kleuren (16 miljoen) gemengd worden.

Een witte achtergrond bekom je door de RGB kleuren op volle intensiteit te mengen:

```
body {
  background-color: #ffffff;
}
```

Een helrode achtergrond bekom je door enkel rood licht toe te voegen:

```
body {
  background-color: #ff0000;
}
```

Een donkerrode achtergrond bekom je door enkel rood licht toe te voegen, maar niet aan volle intensiteit:

```
body {
  background-color: #990000;
}
```

Een roze achtergrond door naast rood licht in volle intensiteit ook gelijke hoeveelheden groen en blauw licht toe te voegen:

```
body {
  background-color: #ff6666;
}
```

Minder gebruikte, maar ook mogelijke kleurnotaties maken gebruik van percentages of decimale waarden (tussen 0 en 255) van de rode, groene en blauwe kleur.

```
body {
    background-color: rgb(60%, 40%, 0%);
}
body {
    background-color: rgb(153, 102, 0);
}
```

CSS3 voegt een vierde waarde toe die de mate van transparantie (a= alpha) beschrijft. De waarde 0 betekent volledig doorzichtig en de waarde 1 ondoorzichtig. De *rgba*-stijldeclaratie maakt ook gebruik van percentages of decimale waarden.

```
div {
    background-color: rgba(60%, 40%, 0%, 0.5);
}
div {
    background-color: rgba(153, 102, 0, 0.5);
}
```

❖ HSL-model

In CSS3 wordt het HSL-model (hue, saturation, lightness) gepromoot in plaats van het RGB-model. Er wordt dus een waarde toegekend aan de tint, de verzadiging en de helderheid. De basis van het HSL-model is het kleurenwiel dat start op 0 graden met rood en kloksgewijs wijzigt naar geel, groen en blauw om dan terug naar rood te evolueren. De rgb-kleuren zijn telkens 120 graden van elkaar verwijderd (rood = 0 of 360, groen = 120 en blauw = 240). De cmyk-kleuren zitten daar net tussen in (geel = 60, cyan = 180, magenta = 300).

```
body {
    background-color: hsl(0, 100%, 50%);
}
body {
    background-color: hsl(120, 25%, 25%);
}
```

Ook binnen het hsl-model kan transparantie toegevoegd worden door over te stappen op de *hsla*-stijldeclaratie.

```
div {
    background-color: hsla(120, 100%, 30%, 0.5);
}
```

❖ Webveilig kleurenpalet

Aangezien vroeger de beeldschermen slechts 256 kleuren konden weergeven voorziet de webstandaard 216 kleuren die door elke browser en elke computerconfiguratie correct kan worden weergegeven. Tegenwoordig kunnen haast alle computers miljoenen kleuren weergeven zodat het gebruik van webveilige kleuren minder noodzakelijk wordt.

5.2 Tekstopmaak

❖ Het lettertype

Om geldige HTML5-pagina's te maken dien je de tekstopmaak volledig te definiëren met CSS. In de CSS eigenschap font-family kan je meerdere font namen toevoegen, als "fallback" mechanisme. Als de browser de eerste font niet ondersteunt, wordt de volgende font geprobeerd. Start dus met de gewenste font en eindig steeds met een generieke font-family (zoals serif).

CSS	waarde	voorbeeld
font-family	naam van het lettertype	font-family: arial
font-size	xx-small x-small small medium large x-large xx-large larger smaller in cm mm pt px em ex %	font-size: large font-size: 12pt font-size: 90%
font-style	normal italic oblique	font-style: italic
font-variant	normal small-caps	font-variant: normal
font-weight	normal bold bolder lighter 100 200 300 400 500 600 700 800 900	font-weight: 600 font-weight: bold
font	font-style font-variant font-weight font-size / line-height font-family	font: bold italic 12pt/14pt times

```
<span style="font-family: Arial; size: 12px; color: #FF0033">i nhoud</span>
```

```
body {
  font-family: Arial, Verdana;
  font-size: 12px;
  color: #FF0033
}
```

Voor *font-family* moet de juiste benaming van het lettertype meegegeven worden. Let erop dat deze benamingen verschillen voor Windows, Mac en Linux. Het best kan je dus meerdere alternatieven opgeven en de reeks afsluiten door de familienaam van het gewenste font.

De fontfamilies zijn:

- **Sans-serif:** dit zijn fonts zonder mooie franje. Voorbeelden: Verdana, Arial, Geneva, ...
- **Serif:** dit zijn fonts met franjes, zoals Times, Times New Roman, Georgia, CG Times, ...
- **Monospace:** dit zijn niet-proportionele lettertypes waarin elke letter evenveel ruimte inneemt, zoals Courier, Courier New, Andale Mono, ...
- **Cursive:** dit zijn fonts die op een handschrift lijken, zoals Comic Sans, Apple Chancery, ...
- **Fantasy:** dit zijn talrijke fantasievolle lettertypes, zoals Webdings, Batik Regular, Impact, ...

Indien de naam uit meerdere woorden bestaat moeten aanhalingstekens gebruikt worden.

```
body {font-family: Times, "Times New Roman", Georgia, serif;}
```

In bovenvermeld voorbeeld zal dus eerst gecontroleerd worden of Times voorhanden is, is dit niet het geval dan wordt gecontroleerd op Times New Roman, is ook die niet aanwezig dan wordt het Georgia en als geen van deze drie aanwezig is dan kiest de browser het standaardlettertype dat gedefinieerd is voor de serif familie.

Om ook relatief te kunnen werken ten opzichte van een standaard lettertypegrootte dien je deze eerst te bepalen door de grootte op te nemen in de stijldeclaratie van het body-element. In de stijldeclaratie van de volgende elementen kan je hier relatief naar verwijzen door een relatieve eenheid te gebruiken. Indien dan het lettertype van 12 pt naar 14 pt gezet wordt, zal alle tekst in verhouding aangepast worden.

```
body {
  font-size: 12pt;
}
h1 {
  font-size: 1.8em;
}
h2 {
  font-size: 1.5em;
}
```

❖ Opmaak van de tekst

Naast het lettertype, de grootte en de kleur kan je met CSS ook het uitzicht van de tekst volledig naar je hand zetten en ook de verticale en horizontale uitlijning bepalen.

CSS	waarde	voorbeeld
direction	Tekstrichting: ltr rtl inherit.	direction: ltr
letter-spacing	Spatie tussen letters.	letter-spacing: 0.5em letter-spacing: 0.3em
line-height	Lijnhoogte, mogelijk in alle eenheden + %.	line-height: 200% line-height: 250mm
text-align	left right center justify	text-align: justify
text-decoration	none underline overline line-through blink	text-decoration: overline
text-indent	Inspringen van eerste woord, mogelijk in alle eenheden.	text-indent: 25px text-indent: 10pt
text-shadow	kleur, x, y, vervaging	text-shadow: blue 5px 5px 2px
text-transform	capitalize uppercase lowercase	text-transform: uppercase
vertical-align	baseline sub super top text-top middle bottom text-bottom %	vertical-align: top vertical-align: 20%
white-space	Behandeling van witruimten: normal nowrap pre pre-line pre-wrap.	white-space: nowrap
word-spacing	Spatie tussen woorden, mogelijk in alle eenheden (em, ex, px, in, cm, mm, pt, pc).	word-spacing: 0.5em word-spacing: -3px word-spacing: 2cm

Een toffe toevoeging in CSS3 die al vrij goed ondersteund wordt is de *text-shadow*. De eerste waarde geeft de kleur, dan de horizontale en verticale verplaatsing en een waarde voor de vervaging.

```
p {
  text-shadow: brown 5px 10px 3px;
}
```

In CSS3 zijn extra mogelijkheden opgenomen die echter nog niet ondersteund worden, nl. *text-decoration-style* en *text-decoration-color* om de lijnstijl aan te passen en *text-wrap* om het afbreken van lijnen te bepalen.

❖ Invoegen van een lettertype

Met CSS3 kunnen ook lettertypes gebruikt worden die niet op de client-pc geïnstalleerd zijn. De lettertypebestanden in .eot, .otf of .ttf moeten dan op de webserver geplaatst worden en worden geladen met *@font-face*. Internet Explorer ondersteunt enkel eot-bestanden. In *@font-face* moet een *src* bepaald zijn en kan je *font-stretch* (niet ondersteund), *font-style* en *font-weight* instellen.

```
@font-face {
  font-family: mijnLettertype_1;
  src: url('naamLettertype.ttf'),
       url('Sansation_Light.eot'); /* IE */
}
```

5.3 Het Box-model

Elk blokelement kan verder opgemaakt worden door talrijke stijldeclaraties die invloed hebben op de alinea, de witruimte rondom, de marges, de aan- en afwezigheid van een kader en de vorm en kleur hiervan.



De inhoud van het blokelement zit centraal. Om witruimte te bekomen tussen de inhoud en de boord, dient de **padding** verhoogd te worden. Dan kan een kader (**border**) geplaatst worden en kan er gezorgd worden voor witruimte tussen de boord en omringende elementen door een **margin** te bepalen.

Met gecombineerde CSS kunnen de border, de margin en de padding voor elke zijde (top, right, bottom, left) afzonderlijk opgegeven worden of voor allemaal samen.

Alle marges 5em:

```
{ margin: 5em; }
```

Top en bottom 2em, left en right 4em:

```
{ margin: 2em 4em }
```

Clockwise: top 1em, right 2em, bottom 3em, left 4em:

```
{ margin: 1em 2em 3em 4em }
```

Indien slecht één waarde wordt meegegeven (vb: *margin: 2cm*) geldt dit voor alle zijden, dus in dit geval zullen alle zijden een marge hebben van 2 cm.

Indien er twee waarden worden opgegeven (vb: *margin: 2cm 3cm*), dan zal de bovenmarge 2 cm zijn en de rechtermarge 3 cm. De ontbrekende marges krijgen de waarde mee van de tegenovergestelde kant (dus bottom 2 cm en left 3 cm).

Indien alle zijden een waarde meekrijgen, gelden deze waarden in wijzerzin, te beginnen aan de bovenzijde. Dus *margin: 2cm 4cm 3cm 1cm* betekent top=2 cm, right=4 cm, bottom=3 cm en left=1 cm.

Indien drie waarden worden meegegeven geldt eveneens de richting van de wijzers van de klok. De code *padding: 15px 10px 8px* zal ervoor zorgen dat de bovenpadding 15 pixels bedraagt, die aan de rechterrاند 10 pixels, en die aan de benedenrand 8 pixels. De linkerrand, die niet is weergegeven, krijgt ook 10 pixels, gezien de waarde van de tegenoverliggende zijde.

```
#voorbeeldbox {
  width: 300px;
  padding: 20px;
  border: outset;
  margin: 20px;
  float: right;
  background-color: silver;
}
```

```
<p id="voorbeeldbox">Elk blokelement kan verder opgemaakt worden door
talrijke stijldeclaraties die invloed hebben op de alineaering, de witruimte
rondom, de marges, de aan- en afwezigheid van een kader en de vorm en kleur
hervan. </p>
<h1>het Boxmodel </h1> ...
```


Outline	verkort: color style width	outline: black solid 4px
Width	breedte	width: 8cm width: 25px
Height	hoogte	height: 40px
Float	left right	float: right
Clear	left right both none	clear: both

De *outline*-stijl is verschillend van de *border*-stijl, ondanks vergelijkbare werking. De *border* wordt gerekend bij het element en wordt dus meegeteld in de totale breedte en hoogte van de box. De *outline* valt daarbuiten.

Opnieuw zorgt CSS3 voor enkele lang gemiste mogelijkheden zoals afgeronde hoeken en schaduwen. Om een schaduw te verkrijgen dien je de waarden mee te geven in de box-shadow declaratie: kleur, x-verplaatsing, y-verplaatsing, vervaging en verspreiding. Door inset als extra waarde mee te geven komt de schaduw aan de binnenzijde. Je kan meerdere schaduwen samenvoegen door ze te scheiden door een komma.

Met *border-radius* kan elke hoek een afronding meegegeven worden, te starten met de linkerbovenhoek.

CSS	actie	voorbeeld
box-shadow	voegt een schaduw toe	box-shadow: blue 5px 5px 5px
border-radius	maakt afgeronde hoeken	border-radius: 10px 0px;
border-image	afbeelding als rand gebruiken	border-image: url()

Met behulp van deze nieuwe toevoegingen kunnen resultaten bekomen worden die tot voor kort nog onmogelijk geacht werden. De browserondersteuning stijgt dag na dag.



5.4 Lijnen

We kunnen een tekst logisch onderverdelen door middel van koppen, alinea's en lijsten. Een ander element om verschillende delen van een tekst van elkaar te scheiden is de lijn. Hiervoor gebruiken we de `<hr/>`-tag (horizontal rule), die we met behulp van css kunnen aanpassen naar onze wensen.

Met CSS kunnen deze kenmerken ingesteld worden met *width* voor de breedte van de lijn, met *height* voor de hoogte van de lijn, met *(background-)color* voor de lijnkleur en met *text-align* voor de alineaëring.

Omdat de opmaak van een `<hr/>`-tag niet in alle browser hetzelfde wordt getoond (vooral verschillen in color en height tussen verschillende browsers) wordt vaak in CSS eerst alle bestaande borders “verwijdert” en vervolgens een nieuwe border-top toegevoegd (en eventueel border-bottom voor schaduw-effect).

```
hr {
  border: none;
  border-top: 1px solid red;
  border-bottom: 1px solid gray;
}
```

5.5 Afwijken van het standaard gedrag met CSS

Vele HTML-elementen hebben een voorgedefinieerde opmaak. Zo is bijna voor alle elementen bepaald of ze inline- of blokelementen zijn. Soms is het nuttig daarvan af te wijken en een typisch blokelement toch inline te gebruiken en omgekeerd. Het gedrag van een element kan aangepast worden met de *display*-stijl, waarmee je kan kiezen om een bepaald element zich te laten gedragen als een ander element. Dit is een zeer handige stijldeclaratie en wordt veel gebruikt bij navigatiemenu's.

CSS	gedrag
none	geen box
block	blokelement met box errond
inline	inline
inline-block	inline, maar maakt een box
inline-table	inline, maar maakt een box die zich gedraagt als een tabel
list-item	blok en gedraagt zich als een lijstitem
run-in	afhankelijk van de inhoud wordt gekozen voor blok of inline
table	blok en gedraagt zich als een tabel
table-caption	blok en gedraagt zich als een tabelbijschrift
table-cell	blok en gedraagt zich als een tabelcel
table-column	blok en gedraagt zich als een tabelkolom
table-column-group	blok en gedraagt zich als een tabelkolomgroep
table-footer-group	blok en gedraagt zich als een tabelvoetnoot
table-header-group	blok en gedraagt zich als een tabelhoofding
table-row	blok en gedraagt zich als een tabelrij
table-row-group	blok en gedraagt zich als een tabelrijgroep
inherit	erft de eigenschap van zijn parentelement

Erg oude browsers (vooral IE6, IE7 en IE8) doen moeilijk als je de nieuwe HTML5-structuurelementen gebruikt. Aangezien die elementen allemaal blokelementen zijn kan je dat de oudere browser leren door dit in een CSS-bestand op te nemen.

```
article, aside, figcaption, figure, footer, header, hgroup, nav, section {
  display: block;
}
```

6 Lijsten

6.1 Soorten lijsten

Lijsten zijn containers, dus met een begin- en eindtag. Er zijn drie soorten lijsten: genummerde, ongenummerde en definitielijsten. Voor een genummerde lijst maak je gebruik van het `...` (*ordered list*) element. Voor ongenummerde is er de `...` (*unordered list*) tegenhanger. Opsommingen van definities kunnen zeer handig gebeuren met een definitielijst `<dl>...</dl>` (*definition list*).

Binnen genummerde en ongenummerde lijsten worden de items opgebouwd door het `` (*list item*) element. In definitielijsten staat `<dt>` (definition term) voor het te definiëren woord en `<dd>` (definition description) voor de uitleg van het gedefinieerde woord.

❖ Genummerde lijsten (geordende lijsten)

Geordende lijsten zijn lijsten met automatische nummering.

```
<ol>
  <li>item 1</li>
  <li>item 2</li>
</ol>
```

De browser plaatst zelf de cijfers of letters voor de verschillende items. De nummering van de items moet dus niet worden aangepast bij het verwijderen of toevoegen ervan.

```
<body>
<h3>Dit is een genummerde lijst</h3>
<ol>
  <li>item A</li>
  <li>item B</li>
  <li>item C</li>
  <li>item D</li>
</ol>
</body>
```



De ``-attributen `type` en `start` kunnen vervangen worden door stylesheets. Nieuw voor HTML5 is de toevoeging van het attribuut `reversed`, om een aflopende telling te krijgen.

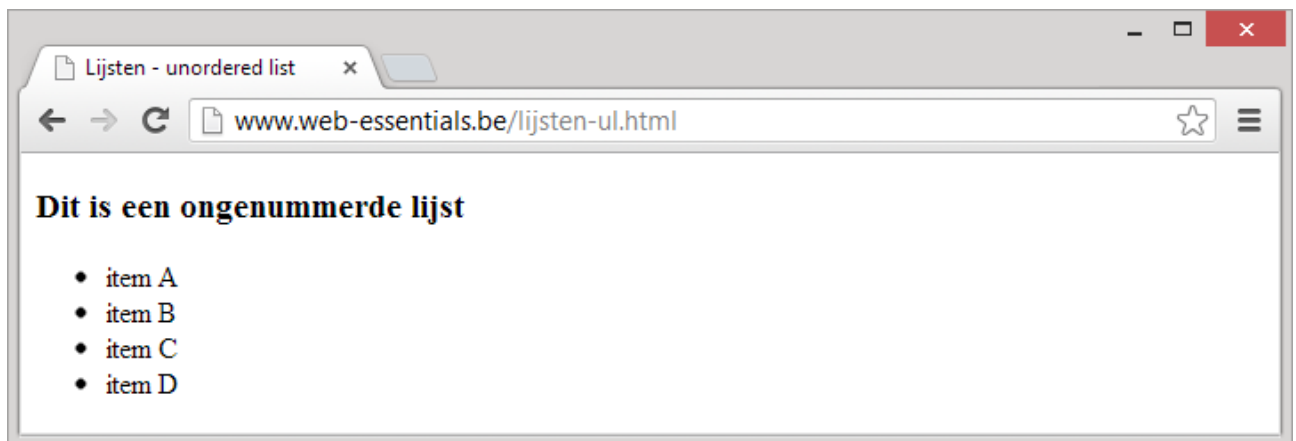
Met CSS kan nummering op *decimal* (1, 2, 3,...), op *lower-roman* (i, ii, iii, iv, ...), op *upper-roman* (I, II, III, IV, ...), op *lower-alpha* (a, b, c, ...) en op *upper-alpha* (A, B, C, ...) gezet worden.

❖ Onge Nummerde lijsten

Ongeordende lijsten zijn lijsten met markeerpunten waarbij de volgorde van items niet van belang is. Op de `` i.p.v. `` na is het gebruik ervan identiek.

```
<ul>
  <li>item 1</li>
  <li>item 2</li>
</ul>
```

```
<body>
<h3>Dit is een onge nummerde lijst</h3>
<ul>
  <li>item A</li>
  <li>item B</li>
  <li>item C</li>
  <li>item D</li>
</ul>
</body>
```



Het enige geldige attribuut van `` is `value`. Het `type`-attribuut van `` en `` is afgekeurd en dus moet je de stijl *list-style* gebruiken (*disc*: ronde zwarte stip, *circle*: open ronde stip, *square*: zwart vierkant).

```
#rondje {list-style: disc}
#cirkel {list-style: circle}
#blokje {list-style: square}
```

```
<ul id="rondje">
  <li>item A</li>
  <li>item B</li>
</ul>
<ul id="cirkel">
  <li>item A</li>
  <li>item B</li>
</ul>
<ul id="blokje">
  <li>item A</li>
  <li>item B</li>
</ul>
```

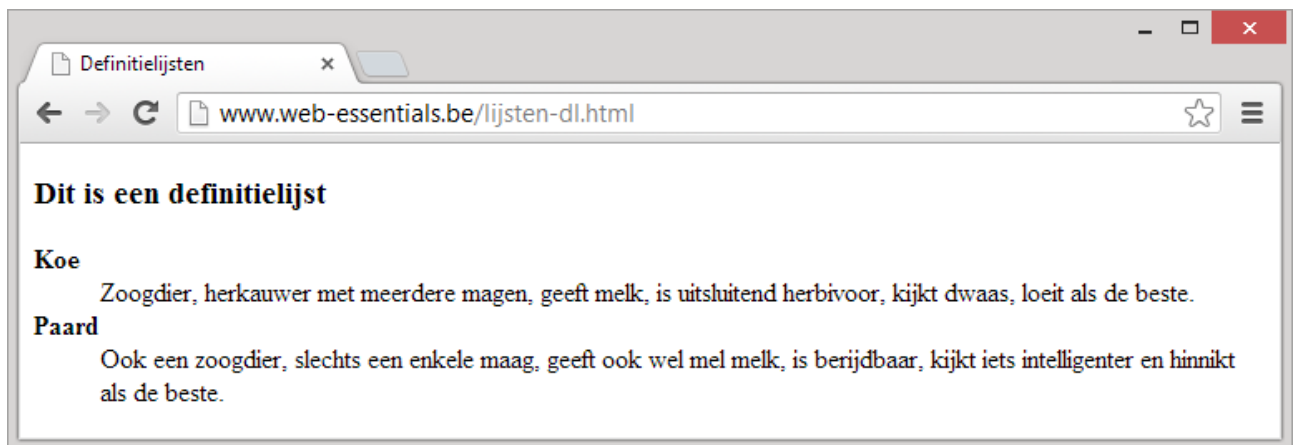


❖ Definitielijsten

Definitielijsten zijn woordenlijsten met telkens een te verklaren term gevolgd door de verklaring. De volledige woordenlijst komt binnen de `<dl>`-container. Elk element uit een definitielijst bestaat uit twee delen: een woord of term `<dt>...</dt>` en de definitie van het woord of de term `<dd>...</dd>`.

```
<dl>
  <dt>term</dt>
  <dd>definitie van woord of term</dd>
</dl>
```

```
<dl>
  <dt><dfn>Koe</dfn></dt>
  <dd>Zoogdier, herkauwer met meerdere magen, geeft melk, is uitsluitend
    herbivoor, kijkt dwaas, loeit als de beste.</dd>
  <dt><dfn>Paard</dfn></dt>
  <dd>Ook een zoogdier, slechts een enkele maag, geeft ook wel melk, is
    berijdbaar, kijkt iets intelligenter en hinnikt als de beste.</dd>
</dl>
```



6.2 Geneste lijsten

Bij het nesten van een lijst binnen een andere lijst springen de verschillende niveaus automatisch in met telkens een ander opsommingsteken (**let op het correct nesten!**). Zet elk nieuw item van een lijst op een nieuwe regel en laat de geneste lijsten inspringen ten opzichte van de hoofdlijst. De lijst is dan in HTML-code veel gemakkelijker te beheren.

```
<body>
<h3>Dit is een geneste
lijst</h3>
<ul>
  <li>item A</li>
  <li>item B
    <ul>
      <li>item A</li>
      <li>item B
        <ul>
          <li>item A</li>
          <li>item B</li>
        </ul>
      </li>
    </ul>
  </li>
  <li>item C</li>
</ul>
</body>
```



6.3 CSS voor lijsten

CSS	waarde	voorbeeld
display	block (gevolgd door een nieuwe regel) inline (niet gevolgd door een nieuwe regel) list-item (met toevoeging van een markeerpunt) none (geen display)	display: inline
list-style-image	url	list-style-image: url(x.gif)
list-style-position	inside outside	list-style-position: inside
list-style	verkorte notatie	list-style: square inside list-style: none list-style: url(y.png) list-style: upper-alpha list-style: lower-roman inside
list-style-type	disc circle square decimal lower-roman upper-roman lower-alpha upper-alpha none	list-style-type: square list-style-type: decimal

Met de CSS-code *list-style-image* kunnen dus eigen opsommingstekens gedefinieerd worden. Hiervoor kunnen alle ondersteunde grafische bestanden voor gebruikt worden.

```
.fig>li {list-style-image: url(figuur.gif);}
```

```
<ul class="fig">
  <li>item</li>
  <li>item</li>
</ul>
```

In combinatie met de CSS-mogelijkheden van fonts en het boxmodel kunnen ogenschijnlijk simpele lijsten omgevormd worden tot mooie menustructuren. Met het *display*-attribuut kan gekozen worden voor een horizontaal of verticaal menu. Met de pseudoklassen kan ook een dynamisch effect bekomen worden.

HTML-code:

```
<nav>
  <ul>
    <li><a href="#">Item 1</a></li>
    <li><a href="#">Item A</a></li>
    <li><a href="#">Item I</a></li>
    <li><a href="#">Item één</a></li>
    <li><a href="#">Item α</a></li>
  </ul>
</nav>
```

CSS-code:

```
nav li a {
  color: white;
  font: normal Verdana, Geneva, sans-serif;
  text-decoration: none;
  list-style-image: none;
  background: #666;
  padding: 5px 30px;
  border: solid 1px black;
  border-radius: 10px 10px 0px 0px;
}

nav li {
  display: inline;
}

nav li a: hover {
  color: #333;
  background: #fff;
  border-bottom: none;
}
```



7 Afbeeldingen

7.1 Bestandsformaten voor het web

Er kunnen verschillende afbeeldingsformaten op een webpagina geplaatst worden. Een veel gebruikt formaat is **GIF** (Graphics Interchange Format) met drie verschillende versies. De GIF87a is bedoeld voor simpele computertekeningen, GIF89a ondersteunt ook transparantie en voor animaties zijn er animated GIF's. Een GIF bevat max. 256 kleuren door zijn kleurdiepte van 8 bits (2^8). GIF's bevatten een ingebouwde compressiemethode zonder kwaliteitsverlies. Dit maakt dat een GIF te verkiezen is voor illustraties en grafieken met vlakke kleuren.

De **JPEG** (Joint Photographic Expert Group) bezit een eigen compressiemethode die erg ideaal is voor foto's en afbeeldingen met veel natuurlijke kleuren door zijn kleurdiepte van 24 bits ($2^{24} = 16,7$ miljoen kleuren). JPEG gooit informatie weg, het compressiemechanisme gaat ervan uit dat het menselijk oog niet zoveel kleuren kan onderscheiden als er in het origineel aanwezig zijn. Eenmaal compressie toegepast, is de weggegooid informatie niet terug te winnen. JPEG zorgt voor een juiste balans tussen hoge kwaliteit en een kleine bestandsgrootte.

Het **PNG** (Portable Network Graphics) formaat ondersteunt 24-bits en 48-bits afbeeldingen. PNG-bestanden zijn meestal groter dan vergelijkbare GIF- of JPEG-bestanden. De compressie is gekenmerkt door minder verlies van informatie in vergelijking met JPEG's. Een duidelijke meerwaarde is dat PNG's transparantie ondersteunen.

	GIF	JPEG	PNG
aantal kleuren	256	16 miljoen	16 miljoen
bestandsgrootte (hetzelfde bestand)	groter	kleiner	kleiner
downloadtijd	langer	korter	korter
decompressietijd	korter	langer	langer
animatie	aanbevolen	afgeraden	afgeraden
transparantie	Gif89a	-	mogelijk
logo's	aanbevolen	afgeraden	afgeraden
computertekening	aanbevolen	afgeraden	afgeraden
foto's	afgeraden	aanbevolen	aanbevolen

Indien transparantie nodig is, dienen de beelden bewaard te worden als GIF89a. Bij deze versie van het GIF-formaat is het mogelijk één van de kleuren transparant te maken. Deze kleur valt dan weg tegen elke gewenste achtergrond. Enkel geschikt voor eenvoudige afbeeldingen waarin de voorstelling zich duidelijk van de achtergrond onderscheidt. Ook PNG ondersteunt transparantie, maar dit wordt nog niet veel toegepast.

7.2 Het invoegen van afbeeldingen

❖ Het -element

Het plaatsen van een afbeelding kan door middel van de open tag ``. De sluitslash is niet langer nodig in HTML5, maar hier wordt gekozen deze toch aan te houden, maar `` is dus ook geldig. Aangezien het `img`-element een inline element is, kan het best in een container geplaatst worden. Zowel het `src` - als het `alt`-attribuut is verplicht. Het `src`-attribuut geeft aan welke afbeelding getoond wordt.

```

```

Met het `alt`-attribuut wordt een alternatieve tekst mee gegeven die getoond wordt als de figuur niet kan weergegeven worden. Indien niet gewenst kan je altijd een leeg `alt`-attribuut plaatsen. Deze figuren worden dan genegeerd door een spraakbrowser.

De tabel toont de nog goedgekeurde attributen van ``, maar sommigen kunnen ook door CSS bepaald worden:

CSS	Attribuut	
	alt	Hiermee geef je aan dat de browser een alternatieve tekst op het scherm zet als de afbeelding om een of andere reden niet zichtbaar is (verplicht binnen HTML5).
height	height	De hoogte van een afbeelding (in pixels of %).
	ismap	Verwijst naar een server-side image map.
	usemap	Geeft aan dat de afbeelding een client-side image map is.
width	width	De breedte van een afbeelding.

De attributen `align`, `border`, `hspace` en `vspace` zijn niet langer opgenomen in de HTML5-standaard, evenals het `longdesc`- en `lowsrc`-attribuut. Ze worden dan ook bij voorkeur vervangen door stylesheets, nl. *float*, *vertical-align*, *border*, *margin* en *padding*.

CSS	Attribuut	
vertical-align	align	Verticale uitlijning: top, text-top, middle, bottom, text-bottom, baseline, sub, super.
float	align	Horizontale uitlijning (left en right).
border	border	border-style, border-width, border-color.
padding		Voegt witruimte toe tussen afbeelding en kader.
margin	hspace vspace	Voegt witruimte toe tussen kader en andere elementen.

❖ Het `<figure>`-element

Het figure-element is nieuw in HTML5 en is bedoeld om foto's, illustraties, diagrammen, codelistings, enzovoort in te kapselen. Het is vergelijkbaar met het inkapselen van hyperlinks binnen een `nav`-container. Vooral de mogelijkheid om een bijschrift toe te voegen met het `figcaption`-element is handig, zeker als dit over meerdere figuren gaat.

```
<figure>
  
  
  <figcaption>Links beeld1 en rechts beeld2</figcaption>
</figure>
```

❖ Het `<object>`-element

Het `<object>`-element is een containertag `<object>...</object>`. Het wordt gebruikt om objecten in te sluiten in een HTML-document. Dit element is voornamelijk bedoeld voor het invoegen van geluidsfragmenten, filmpjes, animaties, applets, enzovoort (zie hoofdstuk Audio en Video). Het `<object>`-element kan echter ook gebruikt worden om afbeeldingen in te voegen.

Ten tijde van XHTML werd getracht het `<object>`-element te promoten ten koste van het ``-element. Dit is nooit doorgebroken door de ondermaatse browserondersteuning. Binnen HTML5 is van deze vervanging geen sprake meer. Je blijft dus best `` gebruiken.

```
<object data="beelden/beeld1.gif" type="image/gif"></object>
```

De specifieke attributen van `<object>` die aan afbeeldingen gerelateerd zijn:

HTML	waarde
data	Hiermee geef je de locatie aan waar de image (of een ander soort object) zich bevindt.
type	Hiermee geef je het type van het object aan dat werd gespecificeerd in het <code>data</code> -attribuut. Dit attribuut geef je best altijd mee.

	Voorbeelden van MIME-type voor figuren zijn: image/png, image/gif, en image/jpeg
usemap	= "#naam": geeft aan dat de afbeelding een client-side image map is

Voor het <object>-element zijn de attributen width en height ook geldig, maar dat kan je beter met CSS definiëren.

In onderstaande code is voor de linkerfiguur een object-tag gebruikt en voor de rechterfiguur een img-tag. Recente browsers hebben hier duidelijk geen probleem mee. Bij oudere browsers is de ondersteuning niet volledig en worden er schuifbalken getoond rond de figuur.

```
<figure>
  
  <object data="figuren/school.jpg" type="image/jpeg"></object>
  <figcaption>Foto ingevoegd met img-element (links) en met
    object-element (rechts)</figcaption>
</figure>
```



7.3 De positie van de afbeelding

❖ Afbeeldingen en tekst

Wanneer je tekst tussen twee afbeeldingen wil, biedt CSS een mogelijkheid om het afgekeurde align-attribuut te vervangen. Hiervoor is wel wat meer code nodig.

HTML5:

```
<div>
  
  
  <h2>Tussen twee afbeeldingen</h2>
</div>
```

CSS:

```
img          {width: 100px}
img#links    {float: left}
img#rechts   {float: right}
h2           {text-align: center}
```



♦ Tekst en afbeeldingen scheiden:

Een ingevoegde afbeelding beïnvloedt de tekst die erop volgt. In HTML5 dien je de stijl *clear: left*, *clear: right*, *clear: both* en *clear: none* (dit is de standaardwaarde) te gebruiken op het element dat na de figuur komt.

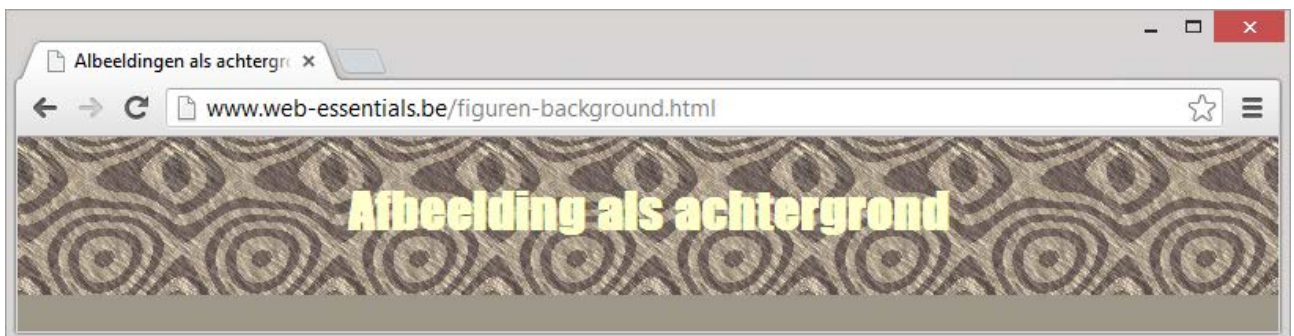
```
h1 {clear: both;}
```

❖ Afbeelding als achtergrond

Binnen HTML5 kan je een achtergrondfiguur plaatsen door gebruik te maken van de stijldeclaraties *background-image:url()* voor de <body>-tag. Dit kan eventueel in combinatie met *background-color*: gebeuren aangezien de achtergrond niet noodzakelijk herhaald wordt tot het browservenster opgevuld is. Indien gewenst kunnen er meerdere achtergrondfiguren opgenomen worden met telkens een andere *background-position*.

```
body { background-image: url (figuren/tegel.jpg);
      background-color: #9E9889;
      background-repeat: repeat-x; }
```

CSS	waarde	voorbeeld
background-attachment	scroll fixed	background-attachment: fixed
background-color	color-rgb color-hex color-name transparent	background-color: red
background-image	url	background-image: url(image.gif)
background-position	·horizontaal (left, center, right) ·verticaal (top, center, bottom)	background-position: center
background-repeat	repeat repeat-x repeat-y no-repeat repeat-x is horizontaal repeat-y is vertical local	background-repeat: no-repeat
background	verkorte notatie	background: red url(image.gif)



Met CSS3 worden de mogelijkheden met achtergrondfiguren behoorlijk uitgebreid. Het nieuwe *background-clip* specificeert het gebied dat de achtergrondfiguur zal innemen. Hierbij wordt de *background-position* relatief bekeken ten opzichte van de padding-box. Met *background-origin* kan je die oorsprong aanpassen. De handigste toevoeging is *background-size*, met auto als standaardwaarde. Je kan de grootte van de achtergrond bepalen door lengtematen op te geven (liefst in pixels), door percentages mee te geven. Met *cover* zal de figuur uitgevuld worden tot de kleinste maat, met *contain* tot de grootste maat. In een vierkante box zullen beide waarden dus geen verschil geven, in een rechthoekige box echter wel.

CSS	waarde	voorbeeld
background-clip	padding-box border-box content-box	background-clip: content-box
background-origin	padding-box border-box content-box	background-origin: border-box
background-size	length percentage cover contain auto	background-size: 800px 600px

Gebruik zoveel mogelijk dezelfde achtergrond op de webpagina's van een site. Het laden gaat sneller omdat de afbeelding na de eerste keer in de cache van de browser staat. Om de leesbaarheid niet in het gedrang te brengen, staat er best geen tekst op de achtergrondaafbeelding. Indien een kleine afbeelding gebruikt wordt die getegeld wordt, kunnen de scherpe randen door middel van filters in een beeldbewerkingsprogramma verzacht worden.

❖ Afbeelding als koppeling

Wanneer je de tag `` opneemt binnen de koppeling-tag `<a>...` werkt de afbeelding als een koppelingstekst. Om de blauwe rand rondom een afbeelding, die als link fungeert, te vermijden, moet je met CSS de boord wegdoen met *border: none*.

```
<a href="..."></a>
```

❖ Image maps

Om aan te geven dat het bij een afbeelding om een image map gaat, moet je aan de ``-tag naast het attribuut `src` ook het attribuut `usemap` toevoegen. Dit is de verwijzing naar het `map`-element met de opgegeven `name`, maar voorafgegaan door een hekje.

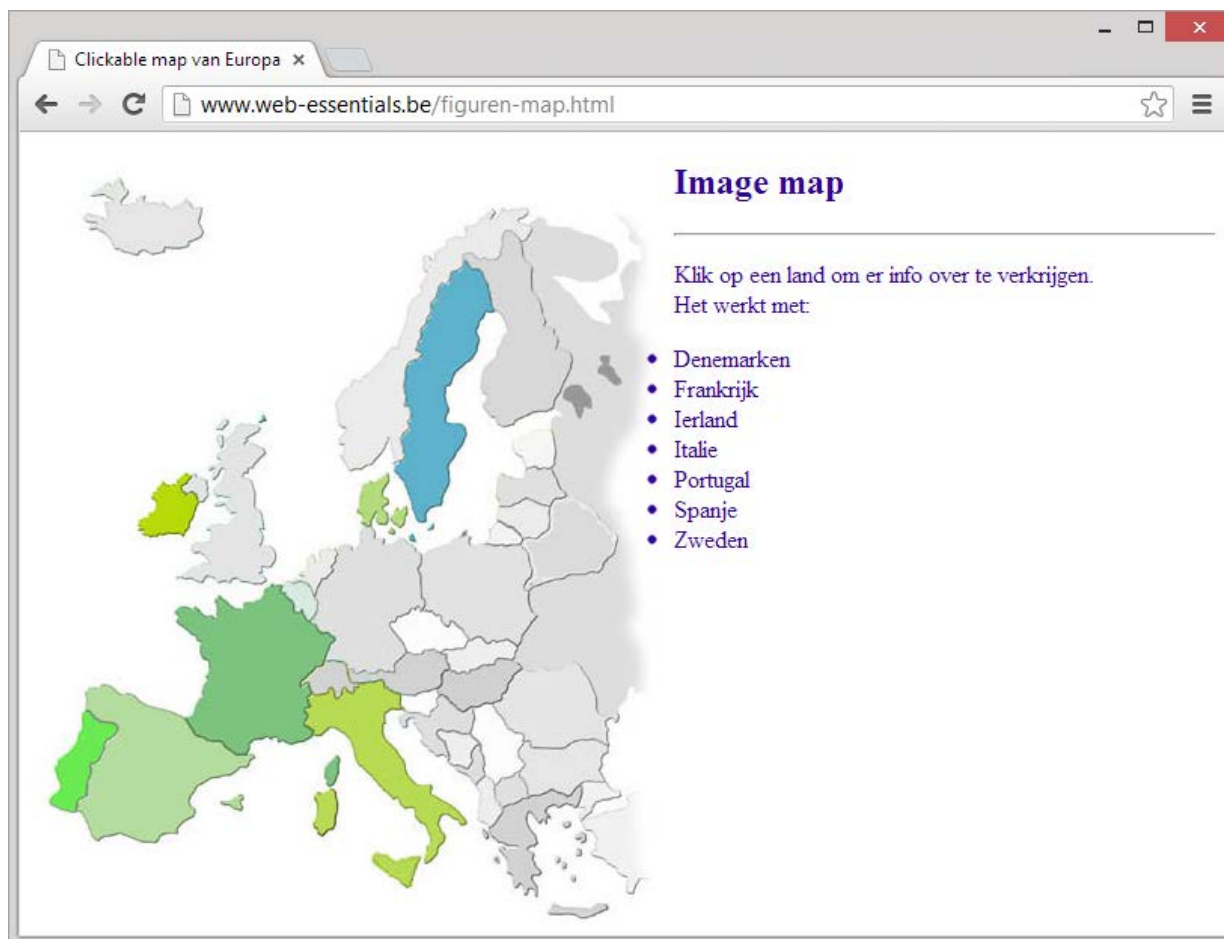
Een `map`-element bevat één of meer `area`-elementen. Elke `area` legt van een aanklikbaar gebied in de afbeelding de vorm, de coördinaten en de bestemming vast.

```

<map name="map">
  <area shape="rect" coords="0, 0, 140, 190" href="beelden/test_1.jpg"
    alt="test 1" />
  <area shape="circle" coords="100, 100, 10" href="beelden/test_2.jpg"
    alt="test 2" />
</map>
```

De attributen van `<area>` zijn:

HTML	waarde
alt	Alternatieve tekst.
coords	Coördinaten, gescheiden door komma's.
href	Hyperlink die geopend wordt.
hreflang	Taal van de hyperlinkreferentie.
media	Geeft het medium aan (vb. screen, pda, ...).
rel	Relatie tussen het document en de doel-url: alternate author bookmark help license next nofollow noreferrer prefetch prev search tag.
shape	Vorm van het gebied: default rect circle poly.
target	Doel waar de link geopend wordt: _blank _parent _self _top.
type	MIME-type van de doel-url.



7.4 Afbeelding klaarmaken voor het web

Bij het invoegen van een afbeelding in een webpagina, wordt best de breedte en hoogte (met *width* en *height* stijlregels) meegegeven. Op deze manier weet de browser hoeveel plaats hij moet openlaten voor de afbeeldingen. Een browser zal immers eerst de tekst laden en daarna pas de figuren. Door gebruik te maken van *width* en *height* kan natuurlijk een figuur ook uitgerekt of samengedrukt worden tot de gewenste afmeting. Dit is echter geen goede werkwijze. Het uittrekken van een figuur gaat gepaard met groot kwaliteitsverlies en bij het samendrukken van grote afbeelding zal deze wel kleiner weergegeven worden op de webpagina, maar blijft de bestandsgrootte echter even groot en dus ook de downloadtijd. Het best worden figuren dus aangepast aan de grootte van de webpagina. De beste werkwijze is ervoor zorgen dat het bestand zo klein mogelijk is en dus weggeschreven wordt in een resolutie waarmee de figuur op het scherm getoond wordt. Maar niet enkel het aantal pixels bepaalt de downloadtijd, ook het aantal kleuren heeft een invloed.

De onderstaande tabel geeft een idee van de gemiddelde downloadsnelheid van een 150 KB bestand met verschillende soorten modems (hier is rekening gehouden met een overhead van 30%, wat op zich zeer laag is).

Snelheid	Verbinding	Downloadtijd UU:MM:SS
28.8 Kbps	Analoge modem	00:00:54
56.6 Kbps	Analoge modem	00:00:27
64 Kbps	ISND	00:00:24
128 Kbps	ISND-2	00:00:12
1 Mbps		00:00:01
4 Mbps	ADSL	375 ms
10 Mbps	Kabel	150 ms

❖ Bestandsgrootte

Er zijn meerdere factoren die een invloed hebben op de bestandsgrootte.

◆ *Aantal pixels:*

Met huidige breedbandverbindingen zou je misschien denken dat je niet op de bestandsgrootte van je afbeeldingen moet letten. Een eenvoudig voorbeeld zal duidelijk maken dat dit wel nodig is. Een foto genomen met je 6 megapixel digitaal fototoestel is ongeveer 2.2 MB groot. Het online plaatsen van deze foto zal met een 4 Mbps verbinding minstens tussen de 5 en 10 seconden duren (in vrij ideale omstandigheden). Wanneer echter iemand je pagina met een telefoonmodem wil bekijken, dient hij minstens 6 minuten geduld uit te oefenen en dan spreken we nog maar over die éne knappe vakantiefoto en nog niet over die 41 anderen. Wanneer de bovenvermelde 6 megapixel echter hervormd wordt naar de grootte waarop deze op de webpagina (bijvoorbeeld 640 x 426) wordt weergegeven zal hij echter nog maar 150 KB groot zijn en ligt de downloadtijd terug in de milliseconden. Bekijk beelden tijdens en na bewerking (in een beeldverwerkingsprogramma) steeds op 100%.

◆ *Aantal kleuren:*

Het aanpassen van het aantal gebruikte kleuren in een beeld heeft invloed op de bestandsgrootte.

◆ *De compressie:*

Compressie vermindert de informatie opgeslagen in een beeld. Er bestaan twee soorten compressie. Bij **non-lossy** compressie blijven de pixels intact. Het bestand wordt enkel op een efficiëntere manier op schijf weggeschreven dan simpelweg bij het save van een bestand. Het terug openen van het (non-lossy) weggeschreven bestand geeft eenzelfde resultaat als het origineel. Bij **lossy** compressie wordt de informatie van het beeld herleid tot het minimum en is onherstelbaar verloren na het save. Er is een duidelijk kwaliteitsverlies.

◆ *Extra's:*

Extra informatie opgeslagen in een bestand zoals previews, paden, kanalen, ICC-profielen, file-info, enz... zorgen voor een grotere bestandsgrootte

8 Audio en video

8.1 Geluid

❖ Geluidsbestanden

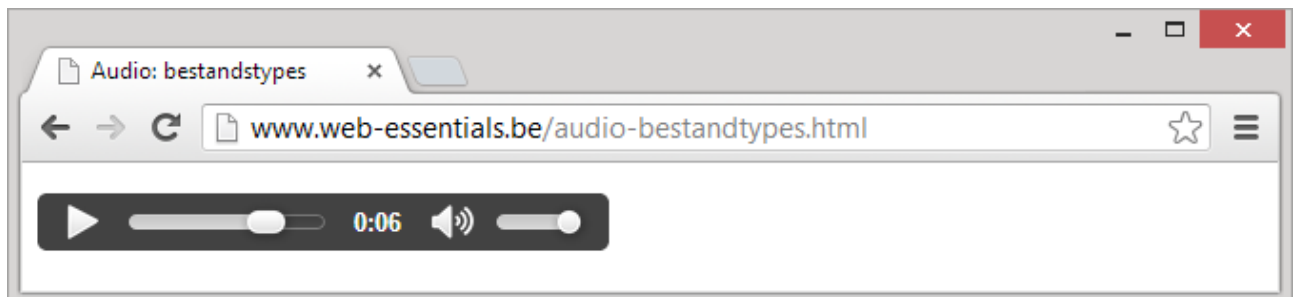
Binnen HTML5 worden er nog drie bestandsformaten voor geluid ondersteund, Ogg Vorbis, MP3 en WAV.

- **WAV:** Het door Microsoft en IBM geïntroduceerde wav-formaat is redelijk eenvoudig van aard. Het nadeel van .wav bestanden is dat ze, in vergelijking met nieuwere formaten, veel ruimte innemen op de harde schijf.
- **MP3:** Het meest gebruikte audio-formaat. Het voordeel van dit formaat is dat het zeer goed geluidsbestanden kan comprimeren. Daarnaast zijn er gratis mp3-players verkrijgbaar waarmee je de nieuwste popsongs kan opnemen en beluisteren. Lange tijd werd er gewerkt met het achtervoegsel .mp2, maar nu is .mp3 de standaard.
- **Ogg Vorbis:** De opensourcemethode om geluidsbestanden te comprimeren. De patentvrije standaard levert een hogere kwaliteit ten opzichte van de oudere MP3-indeling. Ogg Vorbis is het audiogedeelte van de Ogg-standaard. De bestanden krijgen de extensie .ogg.

❖ Geluid invoegen: audio

Met het `audio`-element kan je een geluidsfragment invoegen op je webpagina. Met de attributen `autoplay`, `controls`, `loop`, `preload` en `src` kan je het gedrag bepalen.

```
<audio controls="controls" autoplay="autoplay" src="audio/geluid.wav" />
```



De attributen voor audio zijn:

HTML	actie
<code>autoplay</code>	speelt het geluidsbestand bij het laden van de pagina.
<code>controls</code>	toont de controleknoppen.
<code>loop</code>	speelt het geluidsbestand in een lus.
<code>preload</code>	laadt het geluidsbestand als de webpagina geladen wordt.
<code>src</code>	bronbestand.

❖ Alternatieven: source

Niet alle browsers zijn al klaar voor de drie geluidsformaten. Daarom is het nu aangewezen van je geluidsfragmenten meerdere formaten op de site te plaatsen. Op die manier ben je zeker dat minstens één formaat ondersteund wordt. Door gebruik te maken van het `source`-element kan je de alternatieven opgeven met het `src`-attribuut. Met het `type`-attribuut kan je het juiste MIME-type definiëren.


```
<audio controls="controls">
  <source src="lied.ogg" type="audio/ogg" />
  <source src="lied.mp3" type="audio/mpeg" />
  Je browser ondersteunt geen geluid.
</audio>
```

8.2 Film

❖ Videobestanden

Ook videobestanden zijn beschikbaar in verschillende formaten, voor het gebruik op het web zijn er slechts enkele geschikt. Veel gebruikte extensies zijn avi, mov, mp4 of wmv. Vroeger moest je altijd een plug-in laden om in je browser video te kunnen afspelen. Dit is nu soms nog het geval, maar de nieuwere browsers breiden hun standaardondersteuning voor video snel uit.

De drie meeste gebruikte codecs zijn:

- **WebM** is een open-source bestandsformaat voor video door Google ontwikkeld. Het wordt reeds vrij goed ondersteund ondanks de recente lancering in 2010. WebM bestaat uit de VP8-video-codec en de Ogg Vorbis-audio-codec.
- **MPEG 4 (H.264)**: het mpeg4-videobestandsformaat gebruikt het H.264-compressiealgoritme, net als mp3-audiobestanden. Net als bij audio gaat dit wel ten koste van de kwaliteit. De standaard is ontwikkeld door de Moving Picture Experts Group (MPEG) van ISO/IEC. De codec wordt vaak toegepast in DivX en XviD (MPEG-4-part2) bestanden. De gebruikte extensie is dan .avi of .mov. Ook Apple's Quicktime maakt gebruik van de MPEG4-compressie. Op een Apple-systeem heeft dit formaat de extensie .qt, op het Windows-platform het achtervoegsel .mov.
- **OGG**: de ogg-standaard voor videobestanden is net als bij de audiobestanden open-source. Het formaat wordt onderhouden door de Xiph.Org Foundation. Voor het videoformaat worden andere codecs gebruikt dan bij het audioformaat Ogg Vorbis. Dit is soms verwarrend omdat beide formaten de extensie .ogg meekrijgen. Om die verwarring weg te werken heeft Xiph.Org verzocht om de .ogg-bestandsextensie om compatibiliteitsredenen enkel voor Vorbis te gebruiken. Naar de toekomst stellen zij voor om de extensie aan te passen aan de inhoud, bijvoorbeeld .oga voor audio, .ogv voor video en .ogx voor applicaties. Zowel commerciële als niet-commerciële en zowel vrije als beschermde mediaspelers hebben het Ogg-formaat en de verschillende codecs geïmplementeerd, mede doordat het formaat vrij is.

❖ Video invoegen: video

Het invoegen van video doe je met het video-element. De mogelijke attributen zijn autoplay, controls, height, loop, muted, poster, preload, src en width. De extra attributen ten opzichte van het audio-element zijn vooral bedoeld om het filmvenster aan te passen.

```
<video controls="controls" autoplay="autoplay" src="video/film.webm" />
```

De attributen voor audio zijn:

HTML	actie
autoplay	speelt het filmpje bij het laden van de pagina.
controls	toont de controleknoppen.
height	hoogte van de videospeler.
loop	speelt het filmpje in een lus.
muted	zet het geluidskanaal af.
poster	toont een afbeelding als de video geladen wordt.
preload	laadt het filmpje als de webpagina geladen wordt.
src	bronbestand.
width	breedte van de videospeler.

Net als bij het audio-element kan je meerdere source-elementen opnemen. Door de gebrekkige ondersteuning van de browsers is het momenteel een goed idee om de drie videostandaarden op te nemen.

```
<video controls="controls">
  <source src="film.webm" type="video/webm" />
  <source src="film.mp4" type="video/mp4" />
  <source src="film.ogv" type="video/ogg" />
  Je browser ondersteunt geen video.
</video>
```

8.3 Andere interactieve inhoud

Naast geluid en film kunnen nog meer multimediatekstbestanden geladen worden in een webpagina. Denk dan bijvoorbeeld aan Flash-games, JAVA-toepassingen of Silverlight interfaces. Veel van die toepassingen kunnen momenteel vervangen worden door HTML5-eigen technologieën, zoals canvas. Dit betekent echter niet dat de anderen achterhaald zijn en niet langer ondersteuning nodig hebben.

- **SWF:** Flash (ShockWave Flash) is speciaal voor gebruik op het web ontwikkeld door Adobe en laat toe om interactieve filmpjes te ontwikkelen. Hiervoor is een plug-in nodig (Flash Player).
- **Silverlight:** De Microsofttegenhanger van het flash-formaat, ook bedoeld om complexe grafische animaties te ontwikkelen voor het web. Ook voor Silverlight is een web-plug-in noodzakelijk.

❖ Invoegen van animaties: embed

Het **embed**-element is oorspronkelijk afkomstig van Netscape en was verdwenen in XHTML. In HTML5 is het terug. Het is bedoeld als universele multimediatag waarmee allerlei media kan toegevoegd worden, dus zowel audio, video, animaties, als andere inhoud. Aangeraden wordt om het **embed**-element enkel te gebruiken voor bestanden die niet kunnen geladen worden met het **audio**- of **video**-element. Die twee nieuwe HTML5-elementen genieten dus de voorkeur.

```
<embed src="animatie.swf" type="application/x-shockwave-flash"
width="300" height="300" />
```

Attributen van **<embed>**:

HTML	waarde
height	Hoogte van het multimedia object (in pixels).
src	Het bronbestand.
type	Het MIME-type van het bestand.
width	Breedte van het multimedia object (in pixels).

Het meest nuttig is het **embed**-element voor het laden van flashanimaties en/of als back-up voor het **audio**- en **video**-element.

❖ Invoegen van andere inhoud: object

Net als **embed** kan het **<object>**-element gebruikt worden voor allerlei multimediatekstbestanden (afbeeldingen, audio, video, Java-applets, ActiveX, pdf en flash). De container **<object>...</object>** kan ook gebruikt worden om andere HTML-bestanden in te voegen in je HTML-code.

Een aantal attributen van `object` komt overeen met die van `embed`.

HTML	waarde
<code>data</code>	De locatie waar het object zich bevindt.
<code>form</code>	Het formulier waartoe het object behoort.
<code>height</code>	De hoogte van een afbeelding (in pixels of %).
<code>name</code>	Naam.
<code>type</code>	Specificeert het type van het object dat opgenomen is in het data-attribuut en wordt best altijd meegegeven. Voorbeelden van waardes zijn: "image/png", "image/gif", "video/mpeg", "audio/basic", ...
<code>usemap</code>	Voor client-side clickable maps.
<code>width</code>	De breedte van een afbeelding (in pixels of %).

Dit biedt echter niet dezelfde functionaliteit die `<embed>` biedt. Om aan deze tekortkoming tegemoet te komen kan er gebruik gemaakt worden van opgegeven parameters.

♦ *Het `<param>`-element*

Op de meeste objecten kan je zelf invloed uitoefenen door er parameters aan toe te kennen. Op die manier ben je onder meer in staat om bedieningspanelen uit te breiden, kleuren te veranderen of het lettertype te bepalen. Een parameter voeg je toe met het element `<param>`, dat je plaatst binnen de `<object>`-container.

Het `name`-attribuut is verplicht, met dit attribuut kan je aangeven welke eigenschap je van het object wil aanspreken. Met het `value`-attribuut kan je een waarde toekennen aan de eigenschap die gespecificeerd is bij `name`. Property waardes hebben geen betekenis in HTML; hun betekenis wordt bepaald door het object zelf.

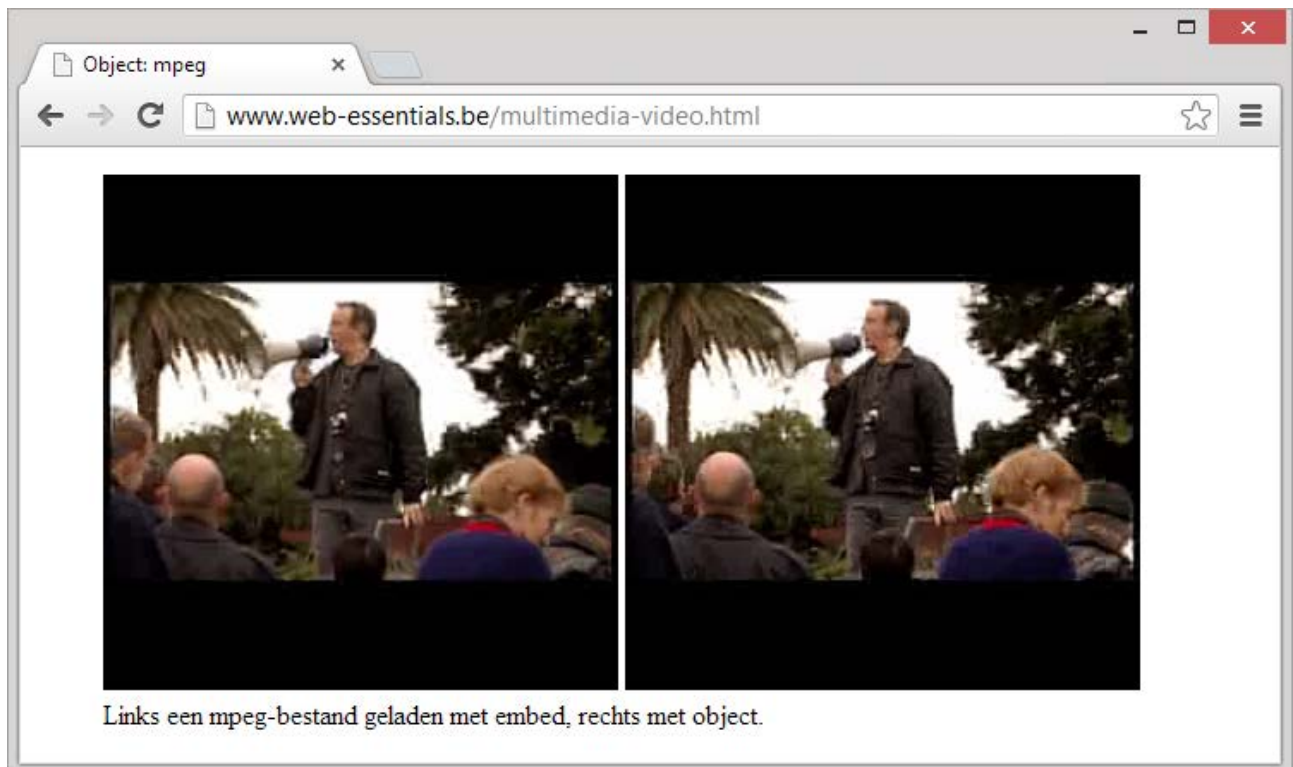
```
<object data="animatie.swf">
  <param name="height" value="40" />
  <param name="width" value="40" />
</object>
```

De tags `<object>` en `<embed>` kunnen in HTML gecombineerd worden als fall-back voor elkaar. Om het gewenste resultaat te verkrijgen in alle browsers is deze combinatie soms toch nodig, eventueel in combinatie met het `audio`- of `video`-element.

```
<object width="550" height="400" data="animatie.swf">
  <embed src="animatie.swf" width="550" height="400" />
</object>
```

Een alternatieve mogelijkheid om `<embed>` volledig weg te laten, maar de bestandsnaam dubbel mee te geven zodat alle browsers er in slagen het bestand te lokaliseren. De eerste maal als `data`-attribuut van het `object`-element en daarna nogmaals als `param`-element met de `name="src"`.

```
<object data="filmpje.mpg" type="video/mpg" width="..." height="...">
  <param name="src" value="filmpje.mpg" />
</object>
```



❖ Invoegen van Youtube-filmpjes

Tot voor kort had je voor het invoegen van Youtube-filmpjes een ingewikkelde code nodig waarin een embed-element werd opgenomen in een object-element. Gelukkig is dit niet langer nodig en wordt er gebruik gemaakt van een iframe-element om video's in HTML5-formaat af te spelen.

```
<iframe width="640" height="385" src="http://www.youtube.com/VIDEO_ID" />
```

9 Tabellen

Tabellen in HTML5 moeten enkel gebruikt worden om gegevens, die in een tabel horen te modelleren. Gebruik een tabel nooit om de lay-out van een webpagina vast te leggen. Daarvoor gebruik je beter <div>, dat is veel flexibeler dan een tabel.

9.1 Structuur van een tabel

❖ De tabel

De volledige gegevens van de tabel worden in de <table>...</table>-container geplaatst. Een tabel wordt rij voor rij opgebouwd, van linksboven naar rechtsonder. Daarna kan elke rij verdeeld worden in de gewenste hoeveelheid kolommen.

Elke zo verkregen cel kan gegevens bevatten. Vervolgens kan je een nieuwe rij toevoegen en herbegint alles.

❖ De tabelrij

Het toevoegen van een nieuwe tabelrij gebeurt met een <tr>...</tr>-container (table row). In deze horizontale rij kunnen meerdere cellen gedefinieerd worden. Indien het aantal cellen in alle rijen gelijk is, komt dit overeen met het aantal kolommen.

```
<table>
  <tr>
  </tr>
</table>
```

❖ De cellen

De tabelrij wordt verder verdeeld in het aantal gewenste cellen met het <td>-element (table data). Alle gegevens die je wilt opnemen in een tabel in een cel komen tussen de codes <td> en </td>. Binnen de tabel en de tabelrij zelf komen dus geen naakte gegevens te staan. Gegevens kunnen bestaan uit tekst, tags voor CSS, lijsten, koppelingen, afbeeldingen of een combinatie van al deze elementen.

```
<table>
  <tr>
    <td>...</td>
    <td>...</td>
  </tr>
</table>
```

De zo ontstane cellen worden met gegevens gevuld.

```
<table>
  <tr>
    <td>GIF</td>
    <td>JPEG</td>
  </tr>
</table>
```

Het element <th>...</th> (table heading) is een variant van <td>, met dat verschil dat de meeste browsers dit herkennen en de inhoud automatisch gecentreerd en vet weergeven. Je kan de eerste rij of eerste kolom van de tabel definiëren als veldnaam door gebruik te maken van de <th>. Dit zijn dan koppen als eerste rij of eerste kolom in je tabel.

Om een goede leesbaarheid te bekomen en snel codefouten te vinden, laat je de codes voor rijen en cellen best inspringen. Nadat de gegevens aangebracht zijn in de cellen kan het resultaat bekeken worden in verschillende browsers. In de ontwerpfase gebruik je best het attribuut border="1" bij het element tabel in de CSS stylesheet, zodat je duidelijk de afmetingen van je tabel ziet. Voor de definitieve versie van je website, kan je dit nog anders opmaken of verwijderen in je stylesheet.

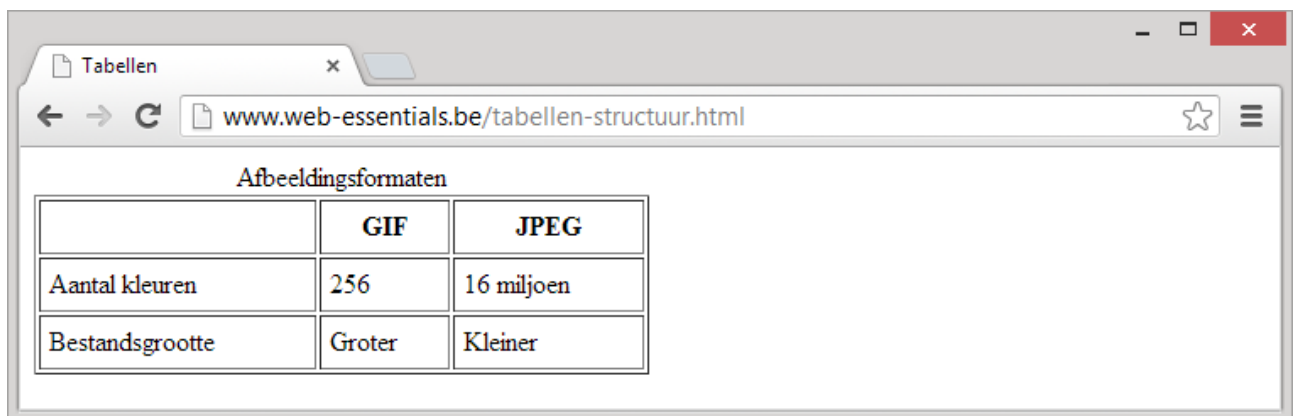
❖ Het bijschrift

Je kan een tabel een titel of bijschrift geven. Hiervoor gebruik je het `<caption>`-element. De standaardpositie van het bijschrift is gecentreerd boven de tabel. Je kan het centreren in CSS aanpassen door `tekst-align(left|right|center|justify|initial|inherit)` en boven of onder de tabel met `caption-side (top|bottom|initial|inherit,)`. Dit wordt ondersteund door alle recente browsers.

```
caption {
    caption-side: top;
}
```

Een voorbeeld:

```
<table border="1">
  <caption>afbeeldingsformaten</caption>
  <tr>
    <th>&nbsp;</th>
    <th>GIF</th>
    <th>JPEG</th>
  </tr>
  <tr>
    <td>Aantal kleuren</td>
    <td>256</td>
    <td>16 miljoen</td>
  </tr>
  <tr>
    <td>Bestandsgrootte</td>
    <td>Groter</td>
    <td>Kleiner</td>
  </tr>
</table>
```



9.2 Attributen

Bij het opmaken van een tabel heb je de keuze of de opmaak geldt voor de gehele tabel, een rij of een cel. Afhankelijk hiervan zal je in CSS de attributen moeten plaatsen bij het juiste element: `table`, `tr` of `td`.

❖ Attributen voor tabellen en rijen

Voor de opmaak van tabellen en rijen (`tr`) zijn niet langer inline attributen ter beschikking. Dit doe je in de CSS stylesheet of in de globale CSS. Het `border` attribuut (`border="1"`) wordt soms wel nog getolereerd tijdens het ontwerpen van je site, als je eigenlijk geen randen wil voor je tabel. Het helpt je om even visueel te zien waar en hoe de tabel gaat komen. Dit attribuut verwijder je voor het online zetten van je website.

❖ Attributen voor cellen

De opmaak van individuele cellen (td of th) ga je ook doen met globale of externe CSS. Enkel om verschillende kolommen of rijen samen te zetten, kan je respectievelijk colspan of rowspan gebruiken. Voor meer uitleg zie verder in dit hoofdstuk.

HTML	waarde
colspan	Overspannen van kolommen (zie verder)
headers	Geeft met behulp van een id een bepaalde relatie tussen cellen aan.
rowspan	Overspannen van rijen (zie verder)

❖ CSS voor tabellen en cellen

Zoals hierboven beschreven is CSS nog het enige geldige alternatief om je tabel op te maken. Samen met de meeste instellingen van het boxmodel zijn er specifieke CSS-stijlen beschikbaar voor tabellen.

CSS	Waarde
border-collapse	collapse (geen ruimte tussen cellen) separate (ruimte tussen de cellen)
border-color	randkleur
border-spacing	witruimte tussen binnen- en buitenrand
border-width	randbreedte
caption-side	plaatsing van het bijschrift
empty-cells	lege cellen tonen of niet: show hide
height	hoogte
padding	witruimte tussen celinhoud en rand
text-align	horizontale uitlijning
vertical-align	verticale uitlijning
width	breedte
table-layout	auto fixed initial inherit

De breedte van de tabel kan je met de stijlregel *width* aanpassen. Om zeker te zijn dat een tabel altijd de volledige breedte van het beeldscherm benut (als alle tekst wordt opgenomen in een tabel) geef je *width:100%* mee. Bij aanpassing van het browservenster wordt de tabel opnieuw opgemaakt. De browser zorgt ervoor dat de tabel nooit te klein wordt voor de inhoud.

```
table {
  width: 100%
}
```

Een tabel die in zijn geheel minder dan 100% van het browserscherm beslaat kan je centreren door de stijlen *margin-left: auto* en *margin-right: auto* mee te geven.

Om de afmetingen van een individuele cel te wijzigen worden de stijleigenschappen *width* en *height* gebruikt. Deze waarden kunnen ook in procenten (relatieve hoogte en breedte) weergegeven worden. Het aanpassen van de hoogte (height) van één cel heeft gevolgen voor alle cellen in die rij. Als je een hele rij in de hoogte wil aanpassen, is het voldoende wanneer de hoogte gespecificeerd wordt in de eerste cel <td> of voor de hele rij <tr>. Voor de breedte (width) geldt dat het aanpassen ervan in één cel gevolgen heeft voor alle cellen in die kolom.

Vermits we dat niet meer inline doen, ga je hier een id of een class aan moeten meegeven, als niet alle rijen even hoog en niet alle kolommen even breed moeten zijn. Indien je een evenwichtige tabel wil maken, kan je dit op het element instellen.

```
td {
```

```
width: 20%;
height: 50px;
}
```

Door middel de stijlregel *padding* kan je de ruimte tussen de celinhoud en de celrand aanpassen.

Door middel stijlregel *border-spacing* kan je de ruimte tussen de cellen onderling vergroten, de cellen zelf veranderen dan niet van grootte.

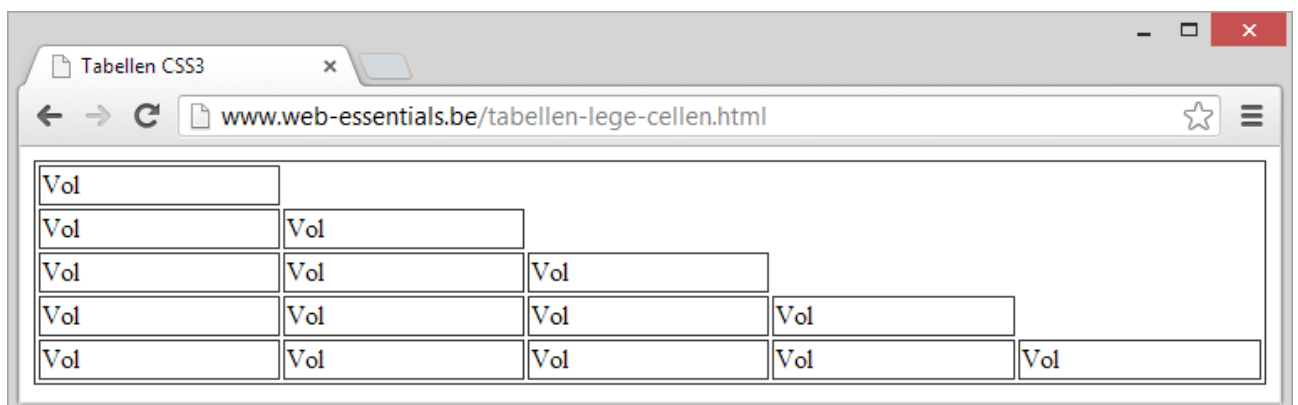
Als je geen ruimte wil tussen de cellen, kan je dit bekomen met *border-collapse*. Dit stijkenmerk overheerst het gebruik van *border-spacing*. Het zorgt er ook voor dat, als je een border tekent, de buitenste celranden met het kader van je tabel één lijn vormen.

Tekst in cellen `<td>` wordt standaard links uitgelijnd, tekst in kopcellen `<th>` wordt standaard gecentreerd. De stijlregel *text-align* (left, right, center, justified) geeft de maker van tabellen de mogelijkheid om qua uitlijning een keuze te maken. Tekst kan je ook verticaal uitlijnen, tekst kan tegen de onderkant of de bovenkant geplaatst worden of juist gecentreerd. De stijlregel hiervoor is *vertical-align* en de opties zijn top, bottom, middle en baseline. De horizontale en verticale uitlijning kunnen ook gecombineerd worden.

```
td {
  text-align: center;
  vertical-align: top;
}
```



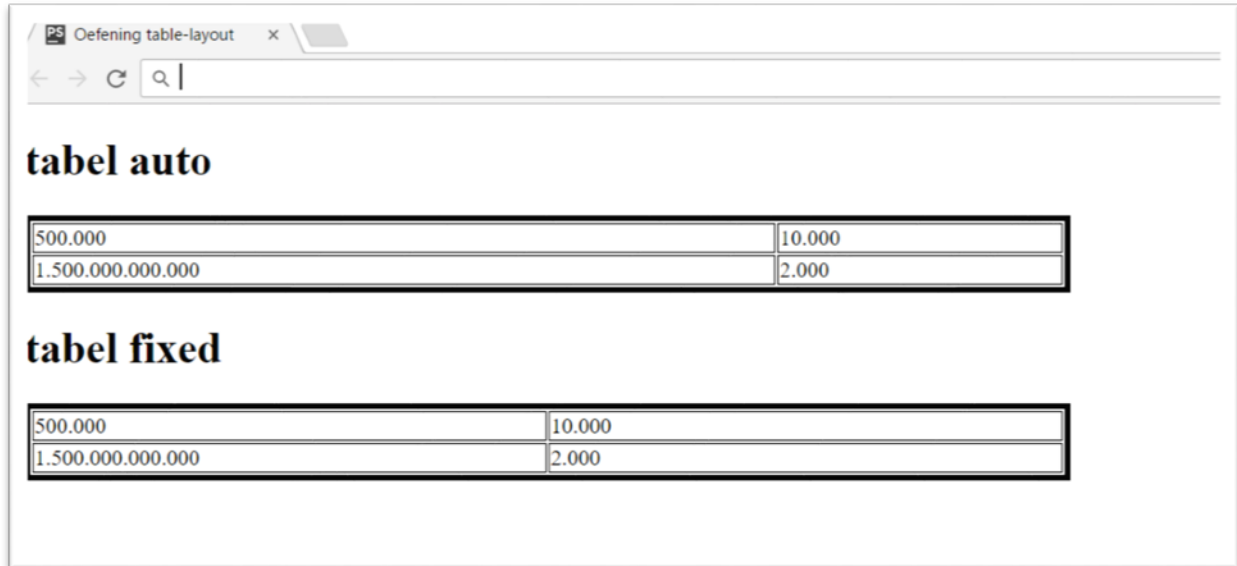
Met *empty-cells* kan je bepalen of lege cellen getoond worden of niet. Het weglaten van lege cellen was voordien onmogelijk.



Met *table-layout* wordt met behulp van een layout algoritme bepaald hoe de tabel getekend wordt. Met de waarde *auto*, wat ook de default is, wordt de kolombreedte bepaald door de grootste

inhoud van een cel. Hierdoor kan dit traag zijn, want er moet eerst door de inhoud van de hele tabel gegaan worden, voordat de finale layout kan vastgesteld worden.

Met de waarde *fixed* is de horizontale layout enkel afhankelijk van de tabelbreedte en niet van de inhoud van de cellen. Dit is sneller dan met *auto*, omdat de browser de tabel kan beginnen tonen, nadat de eerste rij is ingelezen. *initial* stelt de default waarde terug in.



9.3 Rijgroepen en kolomgroepen

❖ Rijgroepen

Om de opbouw van het uitzicht van een tabel te vergemakkelijken kan de tabel ingedeeld worden in drie zones of **rijgroepen**: `<thead>`, `<tfoot>` en `<tbody>` in deze volgorde.

Hiervoor wordt aangegeven welke rijen deel uitmaken van de betreffende rijgroep (hoofd, body, voet). Zo kan je bijvoorbeeld voor alle cellen uit de rijgroep in één keer een aantal stijlen vastleggen (*text-align*, *background-color*, *vertical-align*). Bij gebruik van attributen wordt de opmaak dan doorgegeven naar de betreffende rijgroep.

◆ Tabelkop

Binnen de tabel kan een hoofding gedefinieerd worden door alle rijen die hiertoe behoren te bundelen in een `<thead>...</thead>` container. In de hoofding worden meestal de benamingen van de kolommen aangegeven (= veldnamen). Een voordeel van het werken met `<thead>` is dat een hoofding kan ingesteld worden die op elke bladzijde zichtbaar is wanneer de tabel zich over meerdere bladzijden uitstrekt.

```
thead {
    display: table-header-group;
}
tfoot {
    display: table-footer-group;
}
tbody {
    display: table-row-group;
}
```

◆ Tabelvoet

De tabelvoet `<tfoot>...</tfoot>` werkt analoog aan de tabelhoofding, maar dient om aan te geven welke rijen deel uitmaken van de voet van de tabel. Dit element moet altijd na `<thead>` komen, maar dan mag het zowel voor als na `<tbody>` geplaatst worden. Let wel: elke tabel mag slechts 1 `<tfoot>` element hebben.

◆ *Eigenlijke gegevens*

De <tbody>...</tbody> container dient om aan te geven welke rijen deel uitmaken van de eigenlijke data van de tabel. Indien de tabelbody veel rijen bevat, kan de browser deze voorzien van een scrollmechanisme. Als je dit in je html niet voorziet, zet de browser toch automatisch een tbody element rond de tr's. Hou hier rekening mee, want hierdoor werkt in CSS table>tr niet.

❖ Kolomgroepen

Ook kolommen kunnen gegroepeerd worden door gebruik te maken van <col /> en <colgroup>. Hiermee worden in één keer de kenmerken van alle cellen van bepaalde kolommen gedefinieerd. Kenmerken die voor de gehele kolomgroep moeten gelden kunnen best opgenomen worden in de <colgroup>-tag. Kenmerken die enkel voor bepaalde kolommen bedoeld zijn worden best in de <col />-tag geplaatst.

Het aantal kolommen in een kolomgroep wordt ofwel bepaald door het opnemen van één of meer <col />-elementen tussen <colgroup>...</colgroup> en/of door het span-attribuut waarin je aangeeft hoeveel kolommen samen gezet worden.

CSS:

```
colgroup {
    vertical-align: middle;
}
#groep1 {
    width: 30%;
    background-color: #99CCFF;
}
#groep2 {
    width: 25%;
    background-color: #CCFFFF;
}
#groep3 {
    width: 10%;
    background-color: #CC99FF;
}
table {
    width: 60%;
}
```

HTML:

```
<body>
<table>
  <caption>Samen zetten en opmaak van kolommen met col en colgroup</caption>
  <colgroup>
    <col span="1" id="groep1"/>
    <col span="2" id="groep2"/>
    <col span="2" id="groep3"/>
  </colgroup>
  <thead>
    <tr>
      <th>Product</th>
      <th>Beschrijving</th>
      <th>Categorie</th>
      <th>Euro</th>
      <th>$</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>8 711444 403064</td>
      <td>Moederbord</td>
      <td>Hardware</td>
      <td>€ 195</td>
      <td>$ 204</td>
    </tr>
    <tr>
      <td>8 711444 403065</td>
```

```

        <td>Harde schijf</td>
        <td>Hardware</td>
        <td>€ 125</td>
        <td>$ 131</td>
    </tr>
    <tr>
        <td>8 711444 403066</td>
        <td>Processor</td>
        <td>Hardware</td>
        <td>€ 243</td>
        <td>$ 255</td>
    </tr>
    <tr>
        <td>8 711444 403067</td>
        <td>Grafische kaart</td>
        <td>Hardware</td>
        <td>€ 95</td>
        <td>$ 100</td>
    </tr>
</tbody>
</table>
</body>

```

col en colgroup

Samen zetten en opmaak van kolommen met col en colgroup

Product	Beschrijving	Categorie	Euro	\$
8 711444 403064	Moederbord	Hardware	€ 195	\$ 204
8 711444 403065	Harde schijf	Hardware	€ 125	\$ 131
8 711444 403066	Processor	Hardware	€ 243	\$ 255
8 711444 403067	Grafische kaart	Hardware	€ 95	\$ 100

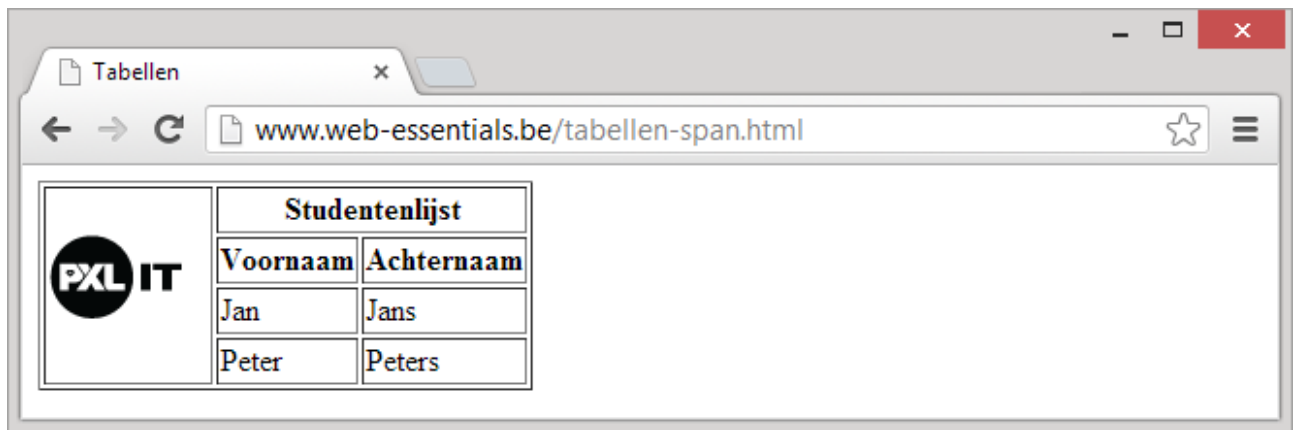
9.4 Overspannen van kolommen en rijen

Soms is het nodig om tabelcellen samen te voegen. Het attribuut `colspan="aantal"` dient voor het overspannen van een aantal kolommen en `rowspan="aantal"` is de tegenhanger voor het overspannen van rijen.

```

<table border="1">
  <tr>
    <td rowspan="4"></td>
    <th colspan="2">Studentenlijst</th>
  </tr>
  <tr>
    <th>Voornaam</th>
    <th>Achternaam</th>
  </tr>
  <tr>
    <td>Jan</td>
    <td>Jans</td>
  </tr>
  <tr>
    <td>Peter</td>
    <td>Peters</td>
  </tr>
</table>

```



9.5 Opeenvolgende tabellen

Men kan probleemloos tabellen na elkaar opnemen.

```
<table>
  <!-- code voor tabel met bijvoorbeeld twee kolommen -->
</table>
<!-- einde van tabel 1 -->
<table>
  <!-- code voor tabel met bijvoorbeeld vier kolommen -->
</table>
```

9.6 Geneste tabellen

Geneste tabellen (tabellen binnen tabellen) zijn ook geen enkel probleem. Binnen de tag `<td>...</td>` van een cel definieer je gewoon een nieuwe tabel die omsloten wordt door de containertag `<table>...</table>`.

```
<table>
  <tr>
    <td>...</td>
    <td>...</td>
  </tr>
  <tr>
    <td>...</td>
    <td>
      <table>
        <tr>
          <td>...</td>
          <td>...</td>
        </tr>
      </table>
    </td>
  </tr>
</table>
```

The screenshot shows a web browser window with the address `www.web-essentials.be/tabellen-genest.html`. The page displays a calendar titled "Activiteitenkalender". It has a header section with three rows: "Vergaderingen" with a single asterisk, "Uitstappen" with two asterisks, and "Meetings" with three asterisks. Below this is a 12-column table for the months of the year. The table content is as follows:

Jan	Feb	Maa	Apr	Mei	Jun	Jul	Aug	Sep	Okt	Nov	Dec
		13*		3**		7***	5**	25**		18	

The table exhibits visual issues: the borders of the header rows and the month header row are not perfectly aligned with the data rows, creating a jagged appearance.

Het mooi aaneensluiten van de randen is wel een probleem, dit kan enkel opgelost worden door de kolom niet te nesten en met kolom- en rijspanning te werken.

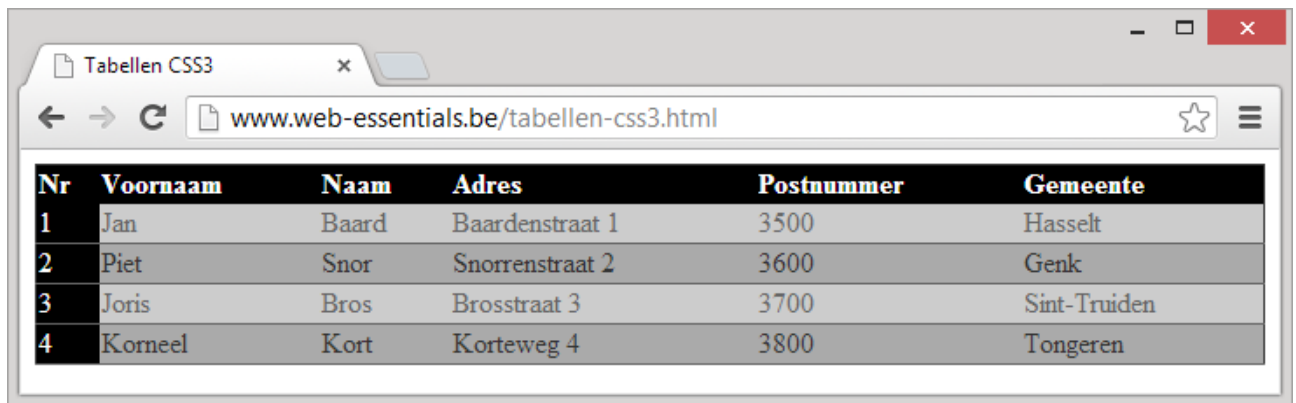
The screenshot shows the same web browser window, but the page is `tabellen-genest2.html`. The calendar table now has perfectly aligned borders across all rows and columns, demonstrating the use of CSS for table styling.

Jan	Feb	Maa	Apr	Mei	Jun	Jul	Aug	Sep	Okt	Nov	Dec
		13*		3**		7***	5**	25**		18	

9.7 CSS3 voor tabellen

De nieuwe pseudoklassen en -elementen bieden zeer handige mogelijkheden om tabellen vorm te geven. Het alternerend vormgeven van rijen of kolommen kan bijvoorbeeld met `:nth-child`, `:first-child` en `:first-of-type`. In plaats van de wiskundige formules te geven kan ook gewerkt worden met `nth-child(odd)` en `nth-child(even)`.

```
table tr:first-of-type {
    background: black;
    color: white;
    text-align: left;
}
tr:nth-child(2n+1) {
    background: #aaa;
    color: #333;
}
tr:nth-child(2n) {
    background: #ccc;
    color: #666;
}
tr > td:first-child {
    background: black;
    color: white;
}
```



The screenshot shows a web browser window with the title 'Tabellen CSS3'. The address bar displays 'www.web-essentials.be/tabellen-css3.html'. The main content area contains a table with the following data:

Nr	Voornaam	Naam	Adres	Postnummer	Gemeente
1	Jan	Baard	Baardenstraat 1	3500	Hasselt
2	Piet	Snor	Snorrenstraat 2	3600	Genk
3	Joris	Bros	Brosstraat 3	3700	Sint-Truiden
4	Korneel	Kort	Korteweg 4	3800	Tongeren

In het hoofdstuk over CSS is er gesproken over het attribuut *display*. Denk er bij de opmaak van je pagina ook aan dat je hier ook voor table (en verschillende varianten hiervan) kan kiezen. Meer hierover bij: https://www.w3schools.com/cssref/pr_class_display.asp

10 Positioneren met CSS

10.1 Waarom CSS positioning

Vanaf de begindagen van het web ervaren ontwikkelaars de nood om een webpagina in te delen in deelvensters. Lang werd dit met frames gedaan. Dit wordt momenteel als een achterhaalde en “old school” techniek beschouwd. Het gebruik van tabellen als alternatief heeft ook zijn nadelen: het vereist zeer veel HTML-code met een slechtere leesbaarheid tot gevolg. Door de grote hoeveelheid en de relatieve complexiteit duurt het ook langer om de pagina met tabellen op te bouwen. Tabellen schenken ook geen absolute vrijheid aangezien alles in een rechthoekige opmaak moet passen en verstoren de semantische opmaak.

Een ander alternatief is om gebruik te maken van layers die zelfs boven elkaar kunnen geplaatst worden. De term layer wordt gebruikt voor elementen die gebruik maken van positioned lay-out, een techniek om elementen te positioneren zonder veel interactie van/met andere elementen. Dit geeft een haast onbegrensde vrijheid, aangezien je deze layers niet alleen exact kan plaatsen, je kan ook alle kenmerken zoals bijvoorbeeld breedte, kleur of zichtbaarheid kan bepalen. Het meest gebruikte voorbeeld is de multikolommenopmaak die toelaat om een pagina er te laten uitzien als een magazine of krant.

De volgende CSS-attributen worden hierbij het meest gebruikt:

CSS	Waarde	Werking
bottom	auto % afstand	De afstand vanaf de benedenrand van de <i>nearest positioned ancestor</i> of van het browservenster.
clear	left right both none	Bepaalt de grenzen van floating elementen.
clip	auto vorm	Bepaalt de vorm van het element.
float	left right none	Bepaalt de positie van het element in haar parent element.
left	auto % afstand	De afstand vanaf de linkerrand van de <i>nearest positioned ancestor</i> of van het browservenster.
overflow	auto hidden scroll visible	Bepaalt wat moet gebeuren als de inhoud groter wordt dan het beschikbare oppervlak.
position	static relative absolute fixed	Plaatst het element op een statische, relatieve, absolute of vaste positie.
right	auto % afstand	De afstand vanaf de rechterrand van de <i>nearest positioned ancestor</i> of van het browservenster.
top	auto % afstand	De afstand vanaf de bovenrand van de <i>nearest positioned ancestor</i> of van het browservenster.
visibility	visible hidden collapse	Bepaalt of een element zichtbaar of onzichtbaar moet zijn.
z-index	auto nummer	Bepaalt de stapelorde.

10.2 Blok en inline-elementen

De HTML-elementen kunnen verdeeld worden in blokelementen en inline-elementen. Elk van deze elementen kan je dan met CSS opmaken. Om een modern ogende webpagina te bekomen zal naast de opmaak ook de positie van elk structuurelement moeten bepaald worden. Dit kan op verschillende manieren: statisch, drijvend, absoluut, relatief en vast.

De standaardinstelling is `position: static`. De lay-out gezien in vorige hoofdstukken gebruikte steeds impliciet deze positionering. In de volgende delen worden de alternatieven besproken.

10.3 Drijvende opmaak

Als je gebruik maakt van de **float**-eigenschap creëer je een drijvende of vlottende opmaak. Wanneer een *float*-waarde wordt meegegeven aan een element bepaalt dit niet enkel de positie van dit element, maar eveneens het gedrag van de andere elementen rond dit element. Indien je een box bijvoorbeeld links plaatst met *float: left* dan zorgt dit ervoor dat de daaropvolgende elementen zich rechts van de box zullen bevinden.

Het is mogelijk om gelijktijdig *float: left* en *float: right* te gebruiken op een pagina. De resterende HTML-code zal zich dan daar tussen wringen. Indien het niet gewenst is dat het volgend element zich rond de box plaatst kan je dit voorkomen met *clear*. Bij *clear: left* zal het volgend element zich onder de links drijvende box plaatsen, bij *clear: right* onder de rechts drijvende box. Indien helemaal geen opvulling gewenst is tussen de boxen gebruik je best *clear: both*.



10.4 Absolute opmaak

Het absoluut gepositioneerde element positioneert zich relatief ten opzichte van haar *nearest positioned ancestor*, haar eerste ancestor element dat niet statisch is gepositioneerd. Als er zo geen ancestor element is, zal het zich relatief ten opzichte van de `<html>`-container positioneren.

Een element met *position: absolute* gedraagt zich als een aparte laag en oefent niet langer een invloed uit op andere elementen, waardoor de volgorde van de absoluut gepositioneerde HTML-elementen helemaal geen rol meer speelt.

In het onderstaand voorbeeld is dit duidelijk. De tekst die na de twee boxen komt, houdt helemaal geen rekening met de aanwezigheid daarvan. Om dit probleem op te lossen kan men proberen alle informatie van de webpagina in een aparte laag te plaatsen.

De positie wordt bepaald door de afstand ten opzichte van rand van het browservenster, omdat er geen gepositioneerde ancestor is, dus met de *top*, *bottom*, *left* en *right* eigenschappen. Bij het wijzigen van de grootte van het browservenster blijft deze afstand ongewijzigd. Een vaste breedte kan meegegeven worden door een *width*-waarde op te geven. Indien je voor alle lagen een vaste breedte instelt zal deze niet wijzigen als het browservenster wijzigt.

Om gedeeltelijk tegemoet te komen aan wijzigingen aan het browservenster kan je ook minimum- en maximumbreedtes en -hoogten definiëren met *min-width*, *max-width*, *min-height* en *max-height*.

```
#box1 {
  position: absolute;
  top: 10px;
  left: 10px;
  width: 200px;
}
#box2 {
  position: absolute;
  top: 10px;
  right: 10px;
  width: 200px;
}
#inhoud {
  position: absolute;
  top: 10px;
  right: 240px;
  left: 240px;
}
```

```
<aside id="box1">
  <p>...</p>
</aside>
<aside id="box2">
  <p>...</p>
</aside>
<article id="inhoud">
  <h1>...</h1>
  <p>...</p>
</article>
```

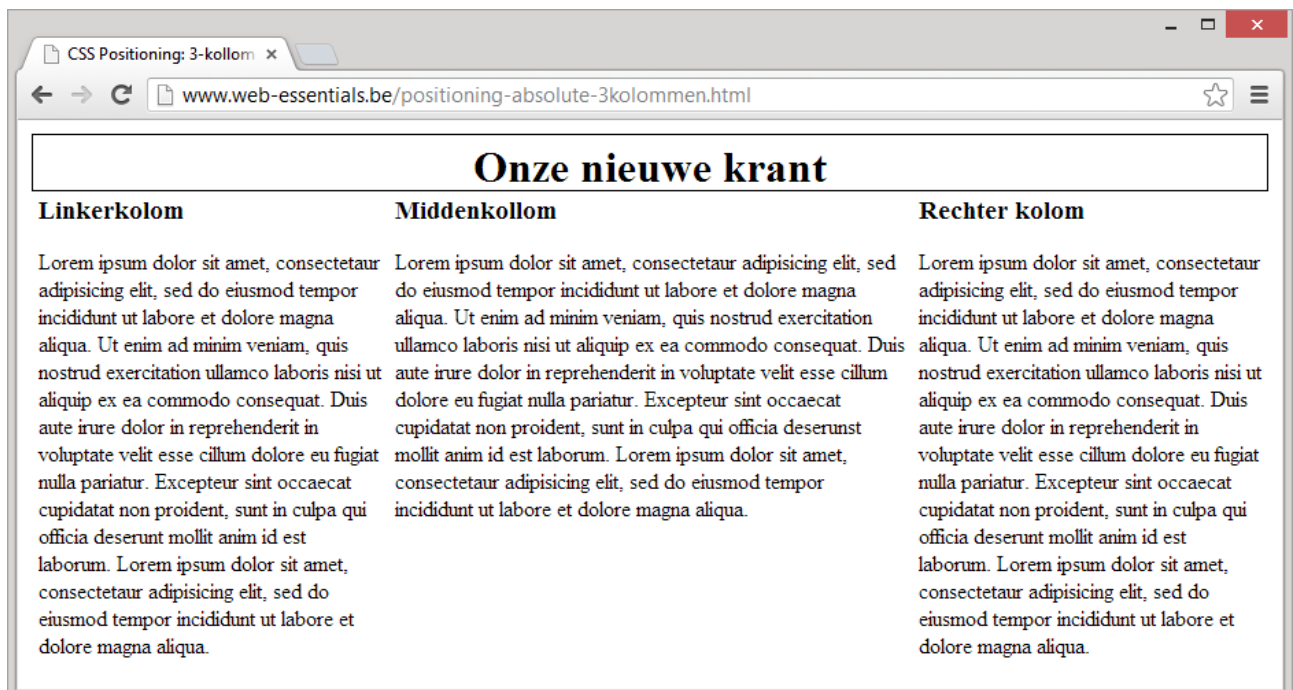


Als je toch het browservenster volledig wenst te benutten kan je van één of enkele lagen de breedte niet meegeven. Om deze laag toch vrij te houden ten opzichte van de andere lagen moet je de afmeting van deze lagen meerekenen in de *top*, *bottom*, *left* en *right* -waarde. Hiervoor moet je rekening houden met alle parameters, dus zowel *width*, *padding* en *margin*.

In onderstaand voorbeeld wordt de *right*- en *left*-waarde van de inhoudlaag ingesteld op 240 pixels. Dit wordt als volgt bekomen: startend van links zijn de eerste 10 pixels afkomstig van de *left*-waarde van box1, de volgende 2 pixels komen van de boorddikte, dan 10 pixels voor de linkerpaddig, dan 200 pixels van de breedte van box1, opnieuw 10 pixels voor de rechterpadding en 2 pixels voor de rechterboord van box1. Dit geeft een totaal van 234 pixels. Aangezien de *left*-waarde van de inhoudlaag op 240 pixels ingesteld staat zijn er dus 6 pixels witruimte tussen beide lagen.



Een extra voorbeeld met een 3-kolommen krantenopmaak illustreert dit beter. Door de linker- en rechterkolom een vaste breedte mee te geven en dit niet te doen voor de middenkolom past deze kolom zich aan aan de breedte van het browservenster.



Indien voor de middenkolom wel een vaste breedte opgegeven wordt zal er een witruimte komen aan de rechterzijde. Om dit te bekomen dien je alle lagen ten opzichte van de linkerkant uit te lijnen. Voor de middenkolom tel je dus van de linkerkolom de 10 pixels linkermarge en de 250 pixels breedte samen. Hierbij tel je dan extra 20 pixels om een witruimte tussen de twee kolommen te bekomen. Voor de middenkolom geeft dit dus *left: 280px*.

Ook dient de titellaag een vaste breedte mee te krijgen door de breedtes van de 3 kolommen op te tellen, samen met de gewenste witruimtes. Om een krantenopmaak te bekomen is voor alle kolommen ook *text-align: justify* meegegeven.

```
#top {
  position: absolute;
  top: 10px;
  left: 10px;
  width: 790px;
  height: 30px;
  padding: 5px;
  border: 1px solid black;
  text-align: center;
}
.kolom {
  position: absolute;
  text-align: justify;
  top: 60px;
  width: 250px;
}
#links {
  left: 10px;
}
#midden {
  left: 280px;
}
#rechts {
  left: 550px;
}
```

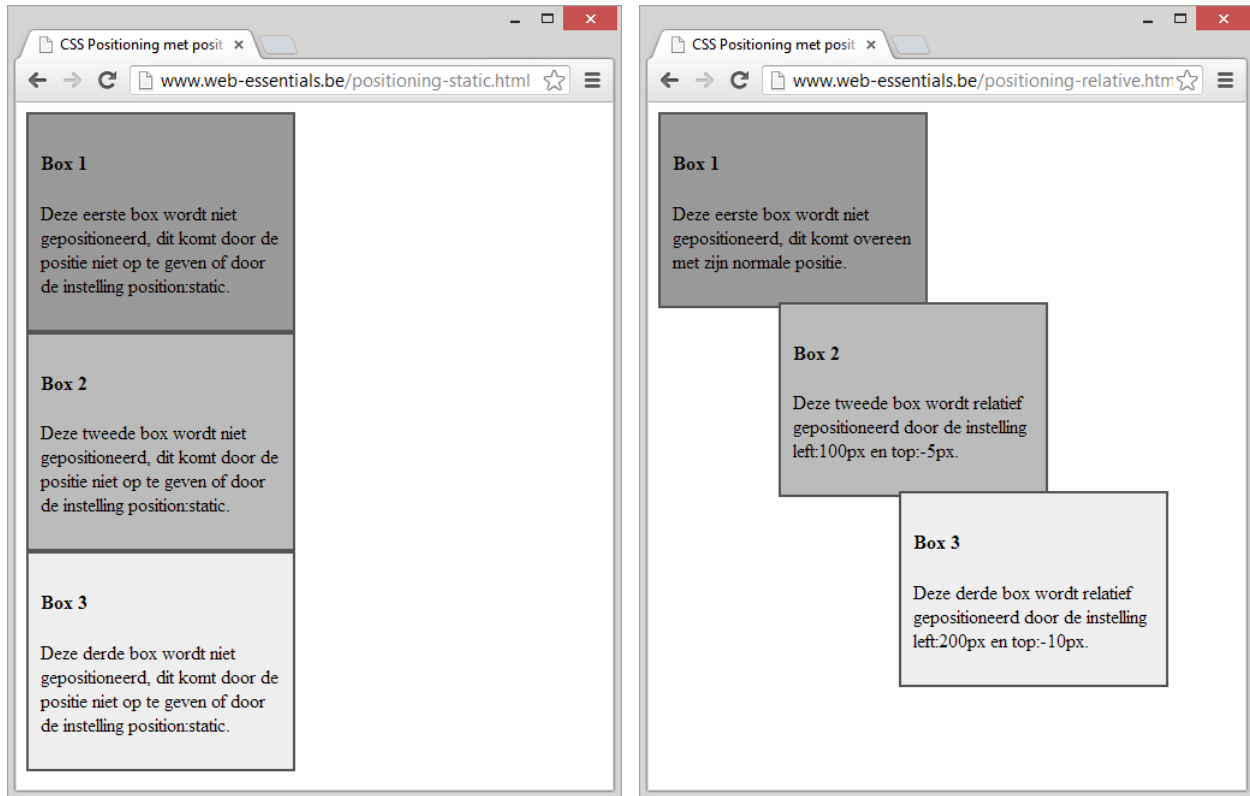


Als alternatief kan ook gebruik worden gemaakt van *right* voor de titel, de rechter- en de middenkolom. Door de titel en middenkolom geen *width* te geven, zullen deze de volledige beschikbare ruimte innemen.

10.5 Relatieve opmaak

Bij gebruik van *position: relative* verhoudt de positie van een element zich relatief ten opzichte van de normale positie die dit element zou hebben zonder positiebepaling (bij de standaardinstelling *position: static*). De normale positie wordt bepaald door de plaats van dit element in de HTML-code. Zo komen opeenvolgende blokelementen allemaal onder elkaar en inline- elementen naast elkaar.

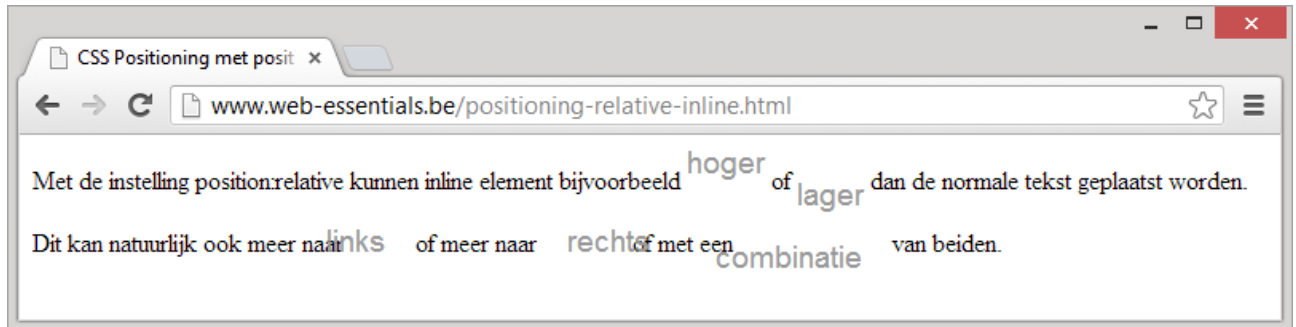
Drie opeenvolgende div's worden normaal perfect onder elkaar weergegeven, maar door het opgeven van *position: relative* en *top*, *bottom*, *left* en *right*, kan deze positie gewijzigd worden.



Bij inline elementen geeft dit ook nieuwe mogelijkheden. Zo kan bijv. een tekstonderdeel verplaatst worden ten opzichte van zijn normale positie.

```
span {
    font-family: Arial, Helvetica, sans-serif;
    color: #999999;
    font-size: 1.2em;
}
.hoger {
    position: relative;
    bottom: 10px;
}
.lager {
    position: relative;
    bottom: -10px;
}
.links{
    position: relative;
    left: -15px;
}
.rechts {
    position: relative;
    right: -15px;
}
```

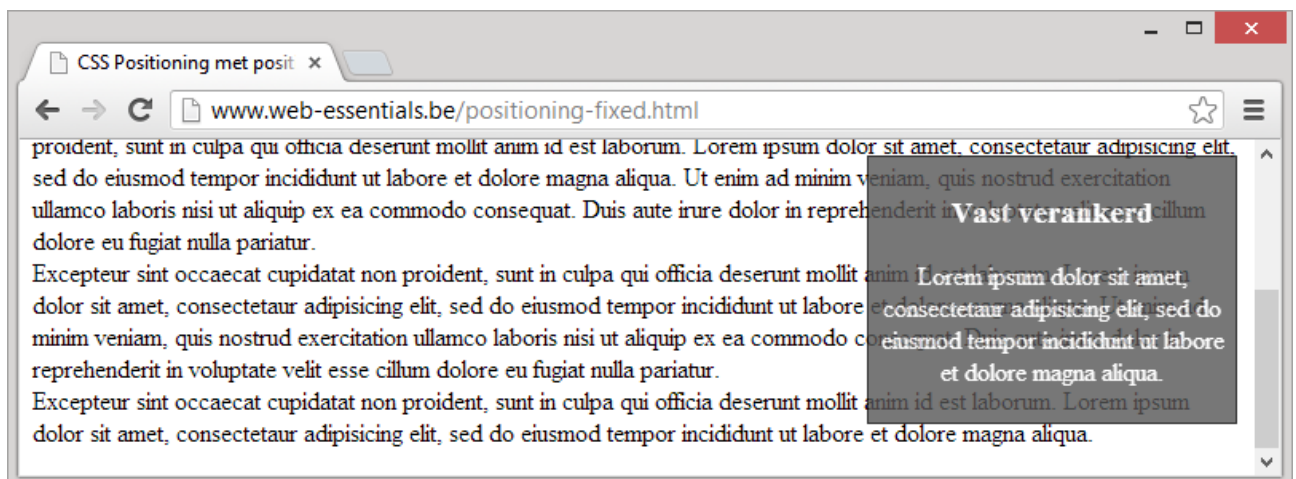
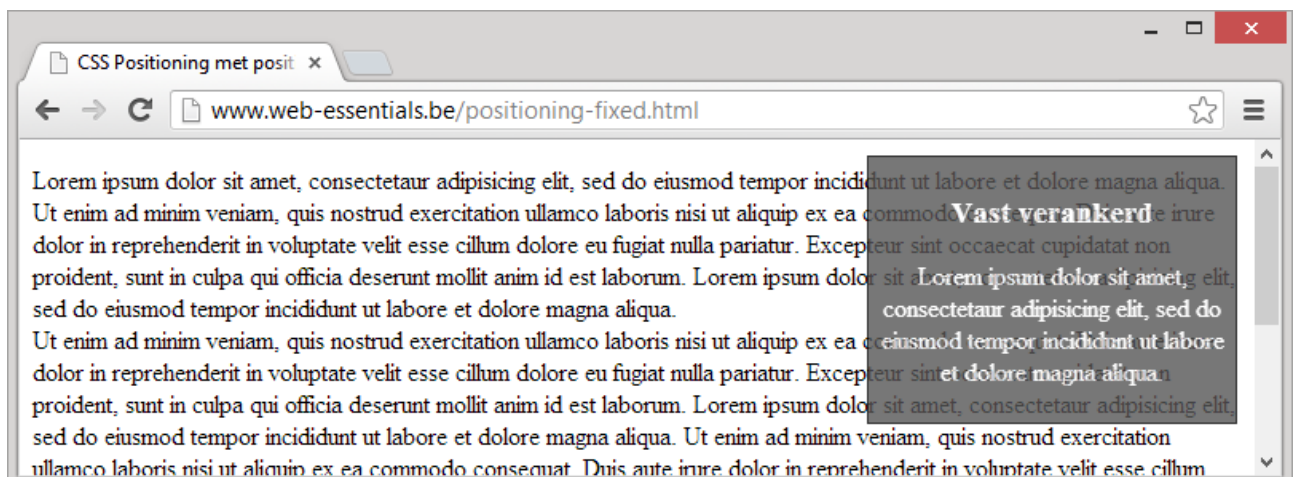
```
<p>Met de instelling position: relative kunnen inline elementen
bijvoorbeeld <span class="hoger">hoger</span> of <span
class="lager">lager</span> dan de normale tekst geplaatst worden.</p>
<p>Dit kan natuurlijk ook meer naar <span class="links">links</span> of
meer naar <span class="rechts">rechts</span> of met een <span class="links
lager">combinatie</span> van beiden.</p>
```



Merk op dat het element nog steeds ruimte in beslag neemt op haar oude positie, alsof het volgens de gewone lay-out niet verplaatst is. Dit is in tegenstelling tot *position: absolute*; waarbij het element uit de normale flow wordt gehaald en geen ruimte meer krijgt toegewezen.

10.6 Vaste positie

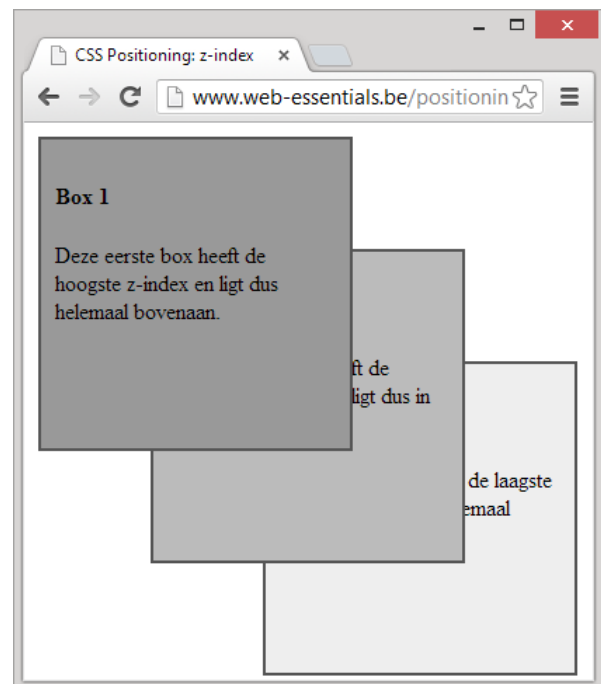
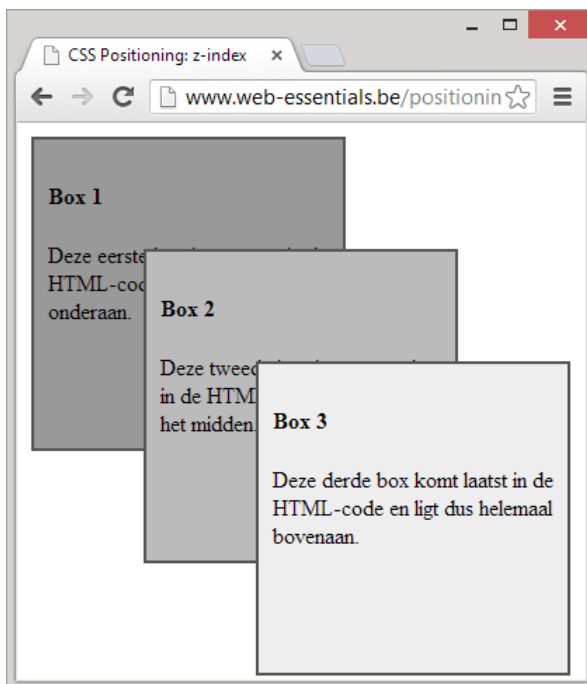
Met de instelling *position: fixed* worden objecten vast verankerd binnen het zichtbare veld van het browservenster. Verder werkt dit identiek aan *position: absolute*. In onderstaand voorbeeld is transparantie gebruikt om het effect te verduidelijken. Let op de positie van de schuifbalk.



10.7 Stapelvolgorde

Zoals in bovenstaande voorbeelden met een absolute en vaste opmaak blijkt, worden de verschillende lagen op elkaar gestapeld. De laag die eerst in de HTML-code staat komt onderaan te liggen en alle volgende lagen zullen hierop gepositioneerd worden in de volgorde waarop ze in de code staan. Dit geeft niet altijd het gewenste effect en het wijzigen van de code levert een verminderde leesbaarheid op. Om manueel in te grijpen op de positie van de lagen kan de *z-index* ingesteld worden en hoeft er niet geraakt te worden aan de volgorde van de code. Objecten met een hogere z-index liggen bovenop objecten met een lager nummer. Er kunnen ook negatieve getallen gebruikt worden.

```
div {
  position: absolute;
  border: 2px solid #555555;
  padding: 10px;
  width: 200px;
  min-height: 200px;
}
#box1 {
  top: 10px;
  left: 10px;
  background-color: #999999;
  z-index: 3;
}
#box2 {
  top: 90px;
  left: 90px;
  background-color: #BBBBBB;
  z-index: 2;
}
#box3 {
  top: 170px;
  left: 170px;
  background-color: #EEEEEE;
  z-index: 1;
}
```



11 Formulieren

11.1 Omschrijving

Formulieren verhogen de interactiviteit van de website en zijn te vergelijken met het dialoogvenster van een programma. Formulieren worden niet enkel gebruikt voor het opgeven van contactgegevens en het versturen van bevestigingen, maar zijn zeker op zeer interactieve pagina's onmisbaar. Bij het klassieke gebruik van een formulier stuurt de websurfer gegevens naar een webserver die daar verwerkt worden door een webapplicatie via een achterliggend script (cgi, php, perl, ...). Die verwerkte gegevens kunnen in een databank bewaard worden en/of desgewenst onder de vorm van een HTML-pagina teruggestuurd worden naar de verzender als respons. De ingevulde gegevens kunnen ook via e-mail verzonden worden. Op interactieve pagina's die bijvoorbeeld simulaties toelaten, worden de gegevens verwerkt door een scripttaal (JavaScript, VBScript, php, ...).

Om ten volle van de meerwaarde van een formulier te genieten moet je dit doordacht opbouwen. Bedenk dus op voorhand goed waarvoor je het formulier nodig hebt en beslis wat er moet gebeuren met de verzamelde gegevens. Het aantal formulieren dat kan opgenomen worden in een HTML-pagina is onbeperkt, maar formulieren kunnen niet genest worden. Indien alle gegevens met één klik van de muis moeten verzonden worden is het wel nodig om alles binnen één formulier te houden.

Formulier:

Naam:

Straat:

Postnummer: Gemeente:

Paswoord:

Land:

Geslacht:

☐ Mannelijk

☐ Vrouwelijk

Operating system:

☐ Windows

☐ Linux

Commentaar:

11.2 Elementen en attributen

Formulieren kunnen zonder problemen in een gewone HTML-pagina opgenomen worden. Ze zijn prima combineerbaar met tekst, afbeeldingen, tabellen, opmaakcodes, enzovoort.

Met formulierspecifieke elementen kan je knoppen, keuzerondjes, keuzelijsten, tekstvakken, paswoorden, enzovoort toevoegen. Aan de gebruiker wordt zo een snelle en efficiënte manier geboden om een website interactief te bezoeken en te verkennen.

❖ Het <form>-element

De container <form>...</form> wordt geplaatst op de plaats in het HTML-document waar je het formulier wil opnemen. De minimale structuur van een formulier:

```
<form>
  Inhoud van het formulier
</form>
```

Om de webserver te vertellen hoe een formulier moet worden verwerkt, gebruik je de attributen `method` en `action`. Het encryptietype geef je aan in `enctype`. Indien een leesbaar ASCII-tekst moet verstuurd worden staat dit op "text/plain".

HTML	waarde
<code>accept-charset</code>	Specificeert de karakterset voor de server.
<code>action="URL"</code>	Hiermee wordt aangegeven wat de server moet doen met de gegevens die je hebt gepost ('post') of die de server heeft opgehaald ('get'). 'URL' geeft het adres aan van het script dat deze gegevens moet verwerken. Door gebruik te maken van een script kan de bezoeker onmiddellijk over de resultaten beschikken.
<code>autocomplete</code>	Schakelt het automatisch aanvullen aan of uit.
<code>enctype</code>	Geeft aan hoe het formulier verzonden wordt naar de server. Met <code>enctype="text/plain"</code> wordt gekozen voor ASCII-tekst (enkel bij <code>method="post"</code>).
<code>method="post"</code> of <code>method="get"</code>	Met 'method' geef je aan hoe de gegevens gecodeerd zullen worden. Met 'post' worden de gegevens niet getoond in de URL, met 'get' worden ze wel getoond.
<code>name</code>	Naam voor het formulier.
<code>novalidate</code>	Geeft aan of veldvalidatie moet gebeuren alvorens te verzenden.
<code>target</code>	Het doel om het resultaat in te tonen.

```
<form action="http://www.domein.com/cgi-bin/..." method="post"
enctype="text/plain">
```

Gebruik een mailto-koppeling als alternatief voor browsers (zeer klein percentage) die geen formulieren ondersteunen. Hier moet `method="post"`, wil je ook data in de mail versturen (=body).

```
<form action="mailto:iemand@domein.com" method="post">
```

Binnen de mailto-koppeling kan ook een onderwerp, een carbon copy en een blind carbon copy opgegeven worden. Deze dienen gescheiden te worden van het mailto-adres door een vraagteken en onderling door een ampersand.

```
<form
action="mailto:iemand@domein.com?subject=onderwerp&cc=iemand. anders@domein
2.com&bcc=nogiemand. anders@domein3.com" method="post">
```


Er moet een mail cliënt geïnstalleerd zijn, opdat dit werkt: de gebruiker moet immers de mail zelf versturen.

11.3 Invoerelementen

De verschillende invoervelden vormen de basisbestanddelen van een formulier. Hiervoor gebruik je het `<input />`-element, waarin het type van invoerveld wordt gedefinieerd met `type=""`. Al vanaf de vroegere HTML-versies kan je met het `input`-element tekstinvoervelden (`text`, `password`, `hidden` en `file`), keuze-elementen (`checkbox` en `radio`), knoppen (`submit`, `reset`, `image` en `button`) maken. In HTML5 zijn er vele types toegevoegd aan het `input`-element (`email`, `url`, `number`, `tel`, `local`, `range`, `date`, `month`, `week`, `time`, `datetime`, `datetime-local`, `search` en `color`). Wanneer de nieuwe input-elementen niet ondersteund worden door de browser, werken ze als een invoerveld van het type tekst. Daarnaast zijn er de elementen `datalist` en `output` bijgekomen.

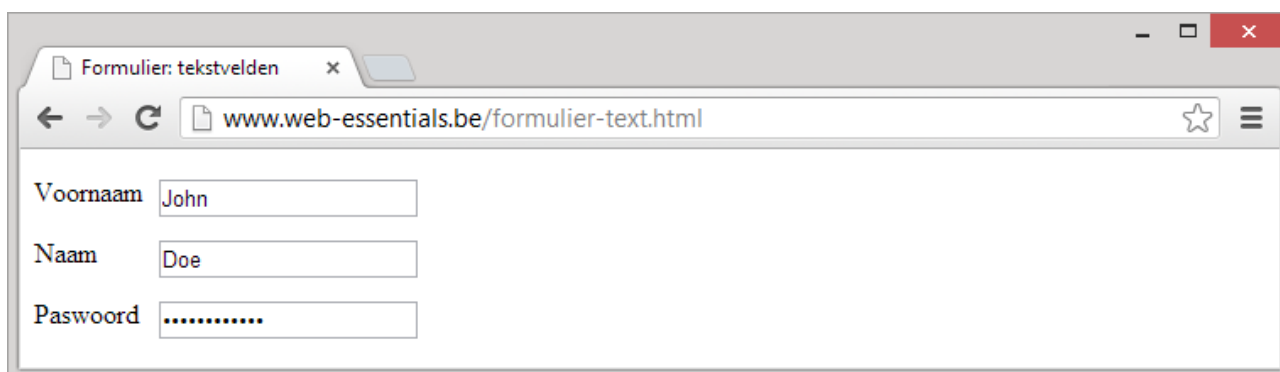
Sommige attributen zijn op bijna alle `<input>`-formulierelementen toepasbaar.

HTML	waarde
<code>autocomplete</code>	Automatisch aanvullen.
<code>autofocus</code>	Automatisch focus krijgen.
<code>disabled</code>	Voorkomt dat de gegevens doorgestuurd worden naar de server.
<code>formnovalidate</code>	Zorgt ervoor dat een element niet gevalideerd wordt.
<code>list</code>	Refereert naar een datalist.
<code>maxlength</code>	Geeft het maximumaantal tekens aan.
<code>name</code>	Zonder dit attribuut stuurt de browser de invoervelden niet door naar de webserver.
<code>pattern</code>	Reguliere expressie.
<code>placeholder</code>	Toont een hint.
<code>readonly</code>	Voorkomt dat de waarde gewijzigd kan worden.
<code>required</code>	Maakt het invoerveld verplicht.
<code>size</code>	Duidt de beginbreedte van het element aan in aantal tekens.
<code>type</code>	Invoertype: <code>button</code> , <code>checkbox</code> , <code>color</code> , <code>date</code> , <code>datetime</code> , <code>datetime-local</code> , <code>email</code> , <code>file</code> , <code>hidden</code> , <code>image</code> , <code>month</code> , <code>number</code> , <code>password</code> , <code>radio</code> , <code>range</code> , <code>reset</code> , <code>search</code> , <code>submit</code> , <code>tel</code> , <code>text</code> , <code>time</code> , <code>url</code> , <code>week</code> .
<code>value</code>	Specificeert de beginwaarde.

❖ Tekstvelden: text

Tekstvelden zijn het meest geschikt voor het verzamelen van algemene informatie. Het is het eenvoudigste veld om als lezer tekstuele en numerieke informatie door te geven. Het invoerveld moet een naam krijgen via het `name`-attribuut. Zonder dit attribuut stuurt de browser de invoervelden niet door naar de webserver.

```
<input type="text" name="..." />
```



De labels kunnen gewoon opgegeven worden, maar kunnen ook door middel van het `label`-element aangegeven worden. Er is geen verschil in weergave, maar er kan wel een functioneel verschil zijn (vb. text-to-speech).

```
<label for="naam">Naam</label><input type="text" id="naam" name="naam" />
```

Met HTML5 kunnen er mogelijkheden voor een invoerveld gedefinieerd worden met een `data list`-container. In deze container plaats je `option`-elementen. Als het element goed ondersteund wordt toont de browser de mogelijkheden. Je kan aan een `label` ook een `value` koppelen. Als de gebruiker dan de `label`-waarde ingeeft, dan komt de `value`-waarde in het invoerveld. De gebruiker blijft altijd de mogelijkheid hebben om toch een vrije waarde in te geven. Het werkt ook als een vorm van autocompletion. Als je enkel `value` voorziet en je begint te typen wordt de keuze mogelijkheid beperkt tot wat past bij wat je typt.

```
Gemeente: <input list="gemeentelijst" name="gemeente" />
<datalist id="gemeentelijst">
  <option label="3500" value="Hasselt">
  <option label="3600" value="Genk">
  <option label="3700" value="Sint-Truiden">
</datalist>
```

❖ Wachtwoordvelden

Wachtwoordvelden werken analoog aan het tekstveld. Het enige verschil bestaat erin dat er in het wachtwoordveld enkel bolletjes verschijnen bij het intikken. Het is enkel een bescherming bij het invoeren van gegevens in aanwezigheid van derden. Alles wat de bezoeker in het wachtwoordveld intikt, wordt echter ongecodeerd over het internet verzonden en kan onderschept worden.

```
<input type="password" name="..." />
```

❖ Hidden velden

Met hidden velden is het ook mogelijk om velden aan te maken die de lezer niet ziet. Deze kunnen dus niet ingevuld worden, maar zijn soms handig om binnen een script te gebruiken.

```
<input type="hidden" name="verborgen veld" />
```

❖ E-mailvelden

Het e-mailveld is bedoeld om het e-mailadres op te vragen. Het is de bedoeling om dan automatisch te controleren of de vormgeving geldig is, met name de aanwezigheid van punten en het `@`-teken. Dit wordt nog niet goed ondersteund, er wordt wel op het `@`-teken gecontroleerd, maar nog niet op punten.

```
<input type="email" name="e-mailadres" />
```

❖ Nummervelden

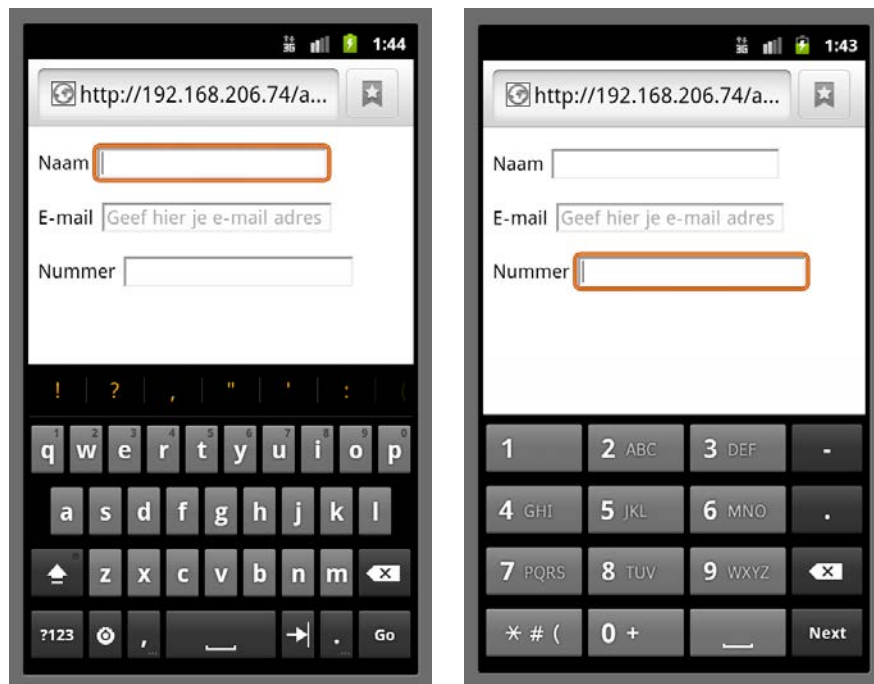
De nummervelden laten toe om een getal in te geven. Met het `min`- en `max`-attribuut kan je de range bepalen.

```
<input type="number" name="nummer" min="0" max="10" />
```

Bij mobiele apparaten zonder toetsenbord wordt vaak automatisch een numeriek toetsenbord gekozen als de focus op dit element komt. Voor andere invoerelementen wordt dan een andere toetsenbordlay-out gekozen. Bij iPhone en iPad werkt dit al voor nummers, e-mailadressen en url-adressen.



De mobiele browsers voor Android-toestellen maken dit onderscheid ook. Ze kiezen een geschikt toetsenbord, afhankelijk van het invoerveld dat de focus heeft.



❖ Tekstinvoer: url

Het `url`-type is een tekstveld dat de vormgeving van een webadres kan valideren en een foutmelding genereert als een incorrecte url wordt ingegeven. Dit is ook enkel HTML5.

```
<input type="url" name="..." />
```

❖ Datumveld: date, month, week, time, datetime, datetime-local

Het invoegen van een vast gedefinieerde datum wordt nog onvoldoende ondersteund, maar zal in de toekomst zeker een meerwaarde zijn als de datum nodig is om berekeningen op uit te voeren om in een databank op te nemen. Er bestaan meerdere datumvelden: `date`, `month`, `week`, `time`, `datetime`, `datetime-local`.

```
<input type="date" name="..." />
```

In Chrome werkt dit reeds: er wordt getoond hoe de datum moet ingegeven worden en als je in het veld klikt, krijg je een kalender.

❖ Zoekveld: search

Het `search`-type werkt hetzelfde als het gewone tekstveld, maar kan een iets andere opmaak meekrijgen (toevoeging van wis-kruis). De bedoeling is om een aparte betekenis te geven aan zoekvelden.

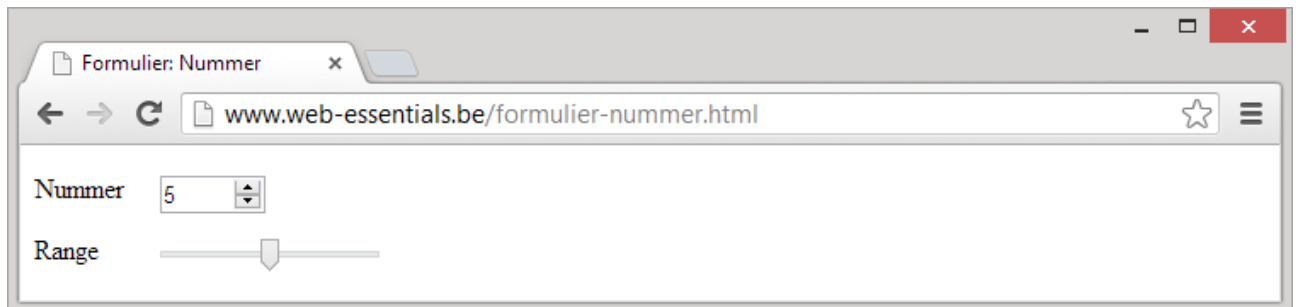
```
<input type="search" name="..." />
```

❖ Range velden

Het range veld kan ook met een minimum en maximum uitgerust worden, eventueel aangevuld met stapsgewijze sprongen.

```
<input type="range" name="range" min="0" max="100" step="10" />
```

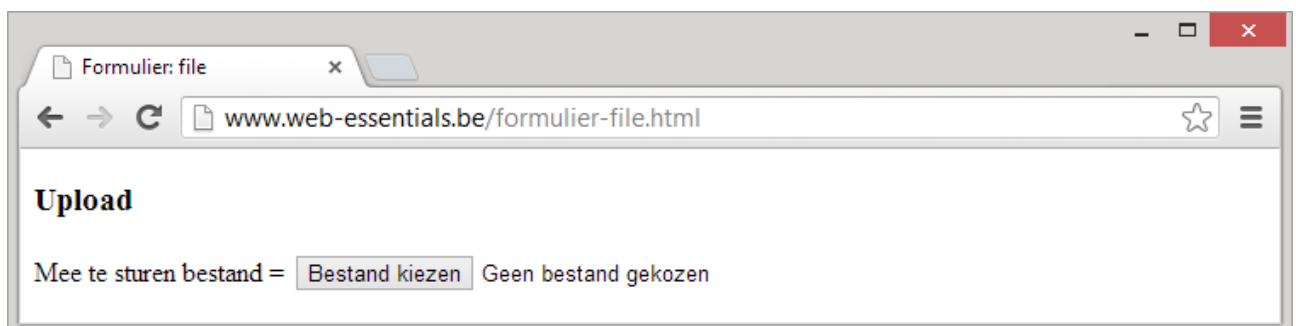
HTML	waarde
max	Maximum waarde
min	Minimum waarde
step	Stapsgewijze wijziging



❖ File velden

Wanneer bestanden dienen verzonden te worden naar de webserver kan je gebruik maken van het `file`-type. Natuurlijk moet de webapplicatie op de server verwachten dat er bestanden (mee)gestuurd worden. De weergave van een file-veld verschilt nogal van browser tot browser, "Bestand kiezen" of "Bladeren"-knop wordt automatisch toegevoegd.

```
<input type="file" name="bij voegsel" />
```



❖ Selectievakken: checkbox

Wanneer de bezoeker een keuze dient te maken uit meerdere mogelijkheden maak je gebruik van selectievakjes. Deze kunnen geactiveerd (voorzien van een vinkje) of gedeactiveerd zijn. Het attribuut `checked` wordt gebruikt om een vakje standaard aan te vinken.

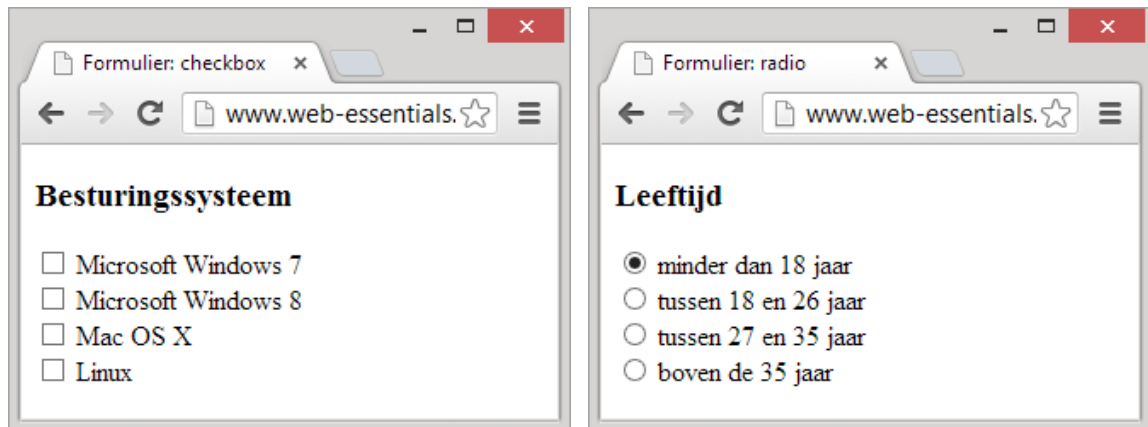
```
<input type="checkbox" name="..." value="..." checked="checked" />
```

❖ Keuzerondjes: radiobuttons

Analoog aan de selectievakjes met dit verschil dat de bezoeker maar één van de mogelijkheden kan selecteren. Dit bereik je door alle keuzerondjes in de lijst dezelfde `name` toe te kennen. Om

toch te weten te komen welk keuzerondje uiteindelijk is geselecteerd, geef je ze allemaal een andere waarde mee (`value="..."`). Deze waarde wordt doorgezonden aan de webserver.

```
<input type="radio" id="leeftijd" name="..." value="..." checked="checked" />
<label for="leeftijd">minder dan 18 jaar</label>
```



❖ Kleurenveld: color

Het `color`-type is nieuw in HTML5 en nog slecht ondersteund. Het laat toe om een kleur te kiezen (soms zelfs met een colorpicker) of een hexadecimale kleurwaarde in te geven.

```
<input type="color" name="..." />
```

❖ Uitvoerveld: output

Nieuw in HTML5 is de logische toevoeging van uitvoervelden bovenop de invoervelden. Het `output`-element is nutteloos zonder een event handler met achterliggend JavaScript (zie Scripting). De meeste browsers ondersteunen dit nieuw element.

```
<form>
  <input type="number" name="getal1" value="0" /> x <input type="number"
  name="getal2" value="0" /> = <output id="product" for="getal1 getal2"
  ></output>
</form>
```

11.4 Keuzelijsten

❖ Dropdownmenu: select

Qua structuur zijn keuzelijsten te vergelijken met genummerde of ongenummerde lijsten. Ze stellen de gebruiker in staat om één of meer items te selecteren uit een menu. Qua functie lijken keuzelijsten op keuzerondjes en selectievakjes. De schermweergave verschilt echter.

Een keuzelijst wordt gedefinieerd binnen de container-tag `<select>...</select>`. De afzonderlijke opties binnen de keuzelijst worden aangeduid door de tag `<option>...</option>`.

De `<select>`-tag kan de attributen `autofocus`, `disabled`, `form`, `name`, `size` en `multiple="multiple"` bevatten.

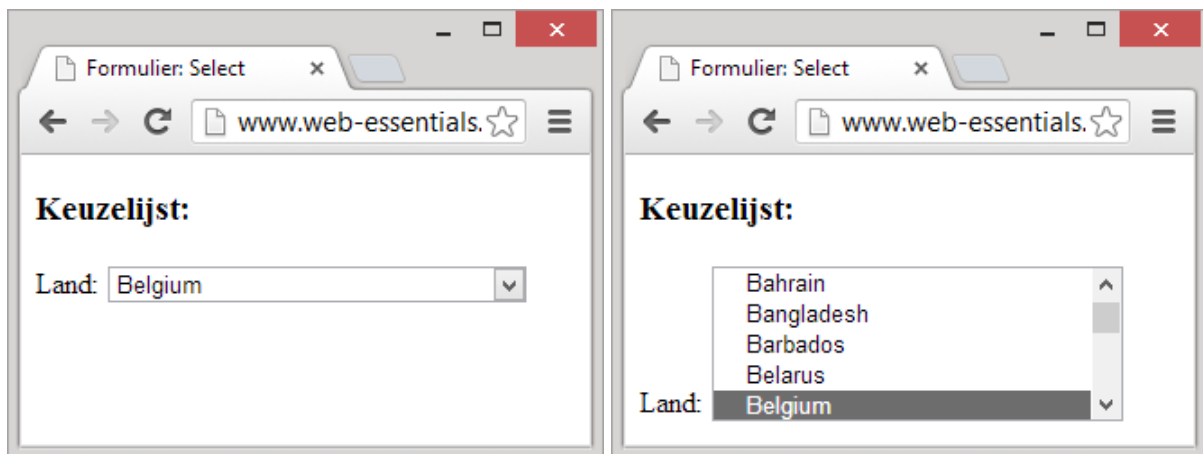
HTML	waarde
<code>autofocus</code>	Zet de focus op de keuzelijst bij het laden van de pagina.
<code>disabled</code>	Zet de keuzelijst op disabled.
<code>form</code>	Definieert de form waar de keuzelijst toe behoort.

<code>multiple="multiple"</code>	Zorgt ervoor dat de gebruiker meerdere items tegelijkertijd kan selecteren door middel van de Ctrl- of Shift-toets in Windows of de Commandtoets bij Mac.
<code>name</code>	Zonder dit attribuut stuurt de browser de invoervelden niet door naar de webserver.
<code>size</code>	Staat voor het aantal items dat de browser effectief toont.

De `<option>`-tag kan de attributen `selected="selected"` en `value="..."` bevatten.

HTML	waarde
<code>selected="selected"</code>	Maakt het mogelijk om een standaardwaarde te selecteren.
<code>value</code>	Specificeert de doorgestuurde waarde.

```
<select name="..." size="..." multiple="multiple">
  <option>...</option>
  <option selected="selected">...</option>
  <option>...</option>
</select>
```



11.5 Tekstvakken

❖ Tekstveld met meerdere regels: `textarea`

Voor het versturen van langere commentaren maak je best gebruik van de `containertag` `<textarea>...</textarea>`. Met de attributen `rows="..."` en `cols="..."` wordt de breedte en hoogte bepaald, uitgedrukt in het aantal karakters. Als je al tekst in het tekstveld wil, kan je die in de container plaatsen.

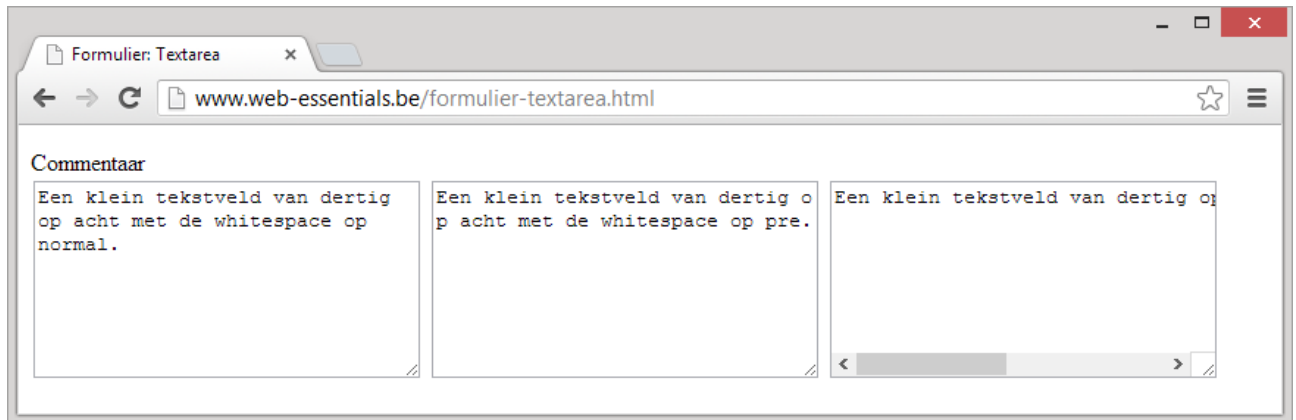
Het al dan niet toevoegen van lijneinden (wrapping) bij het verzenden kan gebeuren met het `wrap`-attribuut dat op `soft` of `hard` gezet kan worden. Bij `soft` wrapping wordt er geen nieuwe regel meegezonden als die er is in het tekstveld, bij `hard` wrapping wordt wel een combinatie van `linefeed` en `carriage return` meegezonden.

Met de CSS-stijl *white-space* kan het uitzicht binnen het tekstveld bepaald worden. De werking verschilt echter per browser.

- *white-space: normal* is de standaardinstelling, de tekst breekt vanzelf af en begeeft zich op de volgende regel als het eind van het tekstveld bereikt wordt.
- *white-space: pre* schrijft de tekst tot de grens van het venster en gaat dan naar een volgende regel, echter zonder af te breken.
- *white-space: nowrap* breekt helemaal niet af en geeft een schuifbalk als de gebruiker blijft doortypen zonder enters in te geven.

```
<label for="commentaar">Commentaar</label>
```

```
<textarea cols="30" rows="8" id="commentaar" wrap="hard"> ... </textarea>
```



11.6 Knoppen

❖ Zend- en wisknop: submit, reset

De waarden die de bezoeker heeft ingevuld, moeten naar de server verstuurd worden. Hiervoor maak je gebruik van een `submit`-knop. Indien je een eigen tekst op de knop wil plaatsen kan dit met het attribuut `value="..."`. Als je geen value meegeeft, komt de standaardtekst op de knop te staan en deze verschilt van browser tot browser. Het indrukken van de submitknop triggert de actie `action="..."` en verstuurt de informatie naar de server. Elk formulier kan maar één submitknop bevatten.

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

Om al de invoervelden in een formulier te wissen gebruik je het `reset`-type:

```
<input type="reset" value="Wissen" />
```

❖ Knop: button

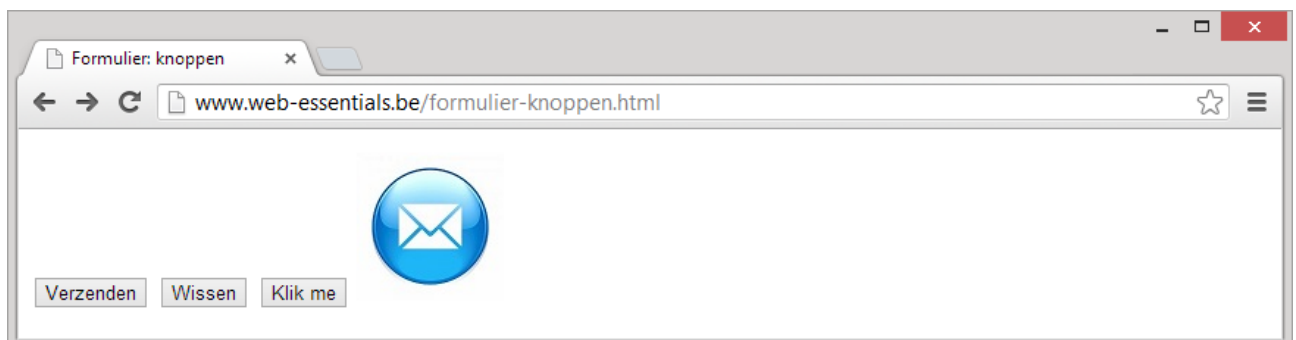
Voor een knop zonder voorgedefinieerde werking en zonder voorgedefinieerd label gebruik je het `button`-type. Hieraan moet je een actie koppelen met een event-handler in JavaScript.

```
<input type="button" value="Klik me" />
```

❖ Figuurknop: image

Als je een figuur als submit-knop wil gebruiken, kan dat met het `image`-type.

```
<input type="image" src="e-mail.jpg" alt="e-mail me" />
```



Voor de submit- en image-knop bestaan er attributen om af te wijken van de in de form gedefinieerde action, enctype, method en target.

HTML	waarde
formaction	overschrijft de standaardactie van de submit of imageknop.
formenctype	overschrijft de standaardencryptie van de submit of imageknop.
formmethod	overschrijft de standaardmethode van de submit of imageknop.
formtarget	overschrijft de standaardtarget van de submit of imageknop.

```
<form action="script.asp" method="get">
  Voornaam: <input type="text" name="voornaam" /><br />
  Achternaam: <input type="text" name="achternaam" /><br />
  <input type="submit" value="Verzenden" />
  <input type="submit" formmethod="post" formaction="postscript.asp"
  value="Verzenden met POST" />
</form>
```

11.7 Structuur in het formulier brengen

❖ Fieldset en legend

Indien bepaalde invoerelementen duidelijk samenhangen kan je best wat structuur aan je formulier toevoegen. Hiervoor is het `fieldset`-element ontwikkeld. Elke `fieldset` kan je een titel geven met het `legend`-element.

```
<fieldset>
  <legend>Persoonlijke gegevens</legend>
  <label for="naam">Naam</label>
  <input type="text" id="naam" name="naam" />
  <label for="voornaam">Voornaam</label>
  <input type="text" id="voornaam" name="voornaam" />
</fieldset>
```

In combinatie met CSS geeft dit een goed gestructureerd formulier.

11.8 CGI-scripts

Een CGI-script (Common Gateway Interface) is een programma dat op de webserver staat en uitgevoerd wordt als reactie op een aanvraag van de gebruiker (door op de submitknop te drukken). Een CGI-script is ofwel geschreven in een taal die gecompileerd wordt om op de server te draaien (C, C++, Ada, Java, Visual .Net) ofwel in een taal die door een interpreter wordt geïnterpreteerd op de server (perl, JCL, ...).

Met CGI-scripts kun je krachtige applicaties op het web toepassen. Dankzij dit soort scripts kan bijvoorbeeld de respons via een formulier op een webpagina geautomatiseerd worden. Dit is vooral handig voor het maken van discussieforums en gastenboeken, maar wordt ook veel gebruikt als alternatief van de "mailto"-actie. Ze worden ook gebruikt om snel gegevens op te halen uit grote databanken. Vandaar de toepassing ervan bij zoekmachines.

CGI-scripts draaien op de webserver van een internet provider. Het resultaat wordt meestal weergegeven als een internetpagina. Dat heeft een groot voordeel, want de bezoeker hoeft geen nagelnieuwe browser en/of plug-ins op zijn eigen computer te hebben. Het werk wordt aan de andere kant op de server gedaan.

Voordat je er enthousiast mee aan de gang gaat is het nuttig om eerst uit te zoeken of je van je provider of systeembeheerder überhaupt scripts mag draaien op hun server. Omwille van de veiligheid en de belasting van hun server mag dat meestal niet of alleen tegen extra betaling. Bij de meeste providers heb je de beschikking over een script voor het bijhouden van een gastenboek en het versturen en verwerken van formulieren. Meestal voorzien providers een /cgi-bin directory waarin de beschikbare scripts zijn opgeslagen.

❖ Een voorbeeld

Stel dat je volgend formulier hebt met de volgende HTML-code.

```
<form method="post" action="/cgi-bin/form.cgi">
<pre>
Naam:      <input type="text" name="naam" size="40" /> <br />
E-mail adres: <input type="text" name="e-mail" size="40" /> <br />
Opmerkingen:
<textarea name="comments" cols="40" rows="5"></textarea>
</pre>
<input type="reset" value="Wissen" />
```

```
<input type="submit" value="Verstuur" />
</form>
```

In de `method` plaats je de waarde "post" of "get", afhankelijk van het script dat op de server staat. In `action` plaats je de URL of de locatie van het CGI-script op de webserver. Meestal staan die in de `cgi-bin` directory, dus bijvoorbeeld: `http://www.server.be/cgi-bin/`. Om de informatie die in het formulier staat te kunnen verwerken, moet de webserver wel CGI-scripts ondersteunen.

Het gebruik van CGI-scripts voor het versturen van e-mail kan vele voordelen bieden ten opzichte van de `mailto`. Er kan opgegeven worden welke pagina de lezer te zien krijgt als hij het formulier heeft verstuurd, je kan de waarden vanuit het formulier koppelen aan tekstbestanden, enzovoort.

❖ Verwerking van de inhoud van de formulierinvoer

Als de gebruiker het formulier verstuurt, ontvangt het CGI-script de data als een setje van naam-waardenparen. De naam is telkens gedefinieerd in de `<input />`, `<select>` of `<textarea>` tag, en de waarde is de inhoud (tekst, getallen, keuzes) of de `value` die de gebruiker heeft ingetypt of geselecteerd.

vb: `naam1=waarde1&naam2=waarde2&naam3=waarde3`

In het script kan je dan hier iets mee doen: bijvoorbeeld als je de naam ontvangen hebt, kan je een html pagina terug sturen waarin je boodschap zet, met die naam erin. Bijvoorbeeld: het is een invulformulier voor nieuwe leden. Na verzenden kan je dan een pagina sturen waarin je het nieuwe lid welkom heet en kort nog wat instructies over het lidmaatschap geeft.

11.9 Toegankelijk formulier

Ook mensen met een beperking surfen op het internet. Zij gebruiken hiervoor hulpmiddelen, zoals een screenreader (= schermuitleesprogramma) voor blinden en slechtzienenden. Wat er nog allemaal kan, kan je vinden op de website <http://anysurfer.be/nl>.

Op bovenstaande site vind je ook een checklist om die toegankelijkheid te testen. Maar het is beter om de aanbevelingen die hierin staan al onmiddellijk bij het maken van de website toe te passen.

Als eerste moet je zorgen dat je met de tabtoets logisch door het formulier kan navigeren. Indien je hier niets wijzigt is dat default van links naar rechts en van boven naar beneden. Maar let op met horizontaal floaten van elementen, in dat geval kan het nodig zijn dat je zelf de tabvolgorde manipuleert. Voordat je dit gaat gebruiken, best even deze pagina lezen: <http://anysurfer.be/nl/in-de-praktijk/websites/tabvolgorde>.

Als we nu gaan kijken naar de checklist om een formulier toegankelijk te maken, is dat niet zo moeilijk om rekening te houden met toegankelijkheid.

Probeer ook er voor te zorgen dat een reeks opeenvolgende velden op dezelfde manier kunnen ingevuld worden: met muis of met toetsenbord. Dus niet veld 1 met muis, veld 2 met toetsenbord, veld 3 weer met muis Probeer ze te combineren, zodat mensen met een motorische beperking niet telkens moeten wisselen.

❖ Labels en invulvelden (input) zijn met elkaar verbonden:

- Voeg aan ieder input veld een id-attribuut toe.
- Plaats een label in een label-element.
- Voeg een `for`-attribuut toe in het label, met de waarde van het id-attribuut van het bijbehorende inputveld.

❖ Input zonder label

In dit geval kan je geen koppeling maken. Gebruik hier dan een tooltip: deze voeg je toe met het `title`-attribuut.

Een tooltip wordt getoond als je met de muis beweegt over een invulveld en wordt ook voorgelezen en in braille weergegeven.

❖ **Label bij datumvelden**

Zorg ervoor dat je in het label meegeeft hoe de datum moet ingevuld worden. Bijvoorbeeld: geboortedatum (dd/mm/jjjj). Indien je een datumpicker zou toevoegen, zorg er dan voor dat dit niet de enige manier is om de datum in te geven, maar dat je hem ook nog gewoon kan typen.

Indien je type=date gebruikt, zul je merken dat je of zelf kan typen, of gebruik kan maken van een datumpicker. (=Chrome)

Omdat het type date nog niet door elke browser ondersteund wordt, kan je ook zelf het datumveld opbouwen, waarbij je voor dag, maand en jaar telkens een aparte selectielijst maakt.

❖ **Samenhorende formulervelden zijn ingesloten door een fieldset.**

Om duidelijk aan te geven dat bepaalde invoervelden (zoals checkboxes, radiobuttons ...) samen horen, is een label niet altijd voldoende. Beter gebruik je hier ook telkens een fieldset, in het legend-element zet je dan de omschrijving.

❖ **Verplichte velden in label duidelijk maken**

Maak duidelijk welke velden verplicht ingevuld moeten worden door in het label (verplicht) of (*) erbij te zetten.

❖ **Een duidelijke verzendknop**

Maak geen keuzelijsten met het onchange-event; kan problemen geven voor mensen, die met behulp van de pijltjestoetsen door de keuzelijst navigeren. Als je dan bij aanklikken van een keuze (pijltje) al naar een andere html pagina zou gaan

Formulieren kunnen soms ook verstuurd worden met de entertoets. Voorzie in dat geval toch altijd een verzendknop, voor mensen die het toetsenbord niet kunnen gebruiken.

Als je een afbeelding als knop gebruikt, vergeet dan zeker het alt-attribuut niet, waarin je duidelijk de actie van de knop omschrijft.

❖ **Bij validatie je foutboodschap aanpassen**

Het is aanbevolen om altijd een boodschap te geven, zowel bij fouten als bij een goed versturen van het formulier.

Dit moet met duidelijke boodschappen gebeuren en niet enkel met kleuraanpassingen.

Omdat je hiervoor met een script moet werken is dit hier niet uitgebreid beschreven.

Als je met javascript kan werken, kan je bij anysurfer op zoek gaan hoe je dat best aanpakt.

❖ **Voorbeelden toegankelijkheid:**

Van alle hier aangehaalde punten zijn voorbeelden terug te vinden op de pagina's van de checklist bij <http://anysurfer.be/nl>

Ga ook voor de andere onderdelen van je website eens kijken in die checklist.

12 JavaScript

12.1 Wat is Javascript?

JavaScript is een programmeertaal die gebruikt wordt om HTML-documenten interactief te maken. Een pagina die enkel uit HTML bestaat is statisch. Dit wil zeggen dat de browser de pagina laadt en enkel toont, er zijn geen extra effecten of interacties. De gebruiker kan de pagina lezen en eventueel op verwijzingen klikken. In combinatie met JavaScript wordt een pagina dynamisch: de pagina reageert op input van de gebruiker. Zo kan je bijvoorbeeld controleren of formulieren correct zijn ingevuld, kleine animaties starten als de gebruiker met de muis ergens over een plaats in de pagina beweegt of nieuwe browservensters op commando openen en sluiten.

JavaScript is een zogenaamde **scriptingtaal**. Programma's hoeven niet gecompileerd te worden (zoals bijvoorbeeld een C-programma), maar ze kunnen onmiddellijk worden uitgevoerd. Binnen de browser is hiervoor een component, de vertolker (engels: interpreter), aanwezig die de programma's uitvoert. Het grote voordeel van vertolkte talen zoals JavaScript is dat de programmeur snel kan testen of een programma werkt, gewoon openen in een browser volstaat. Een nadeel is dat fouten in het programma soms moeilijk te vinden zijn, omdat er geen compiler is die foutmeldingen in de broncode aanduidt.

JavaScript werd in 1995 ontwikkeld door de programmeurs van Netscape, maar werd daarna goedgekeurd door de ECMA (European Computer Manufacturers Association) wat een naamswijziging tot ECMAScript tot gevolg had. Momenteel is de taal ISO-gestandaardiseerd door standaard ISO 16262. De huidige versie is JavaScript ECMAScript2015 of ES6.

1. 1995: JavaScript wordt voorgesteld (eerst onder de naam LiveScript)
2. 1997: ECMAScript standaard wordt goedgekeurd
3. 1999: ES3
4. 2000–2005: XMLHttpRequest, a.k.a. AJAX wordt populair op het web: Outlook Web Access (2000), Gmail (2004) and Google Maps (2005).
5. 2009: ES5 komt uit (meest gebruikt op dit ogenblik) met `forEach`, `Object.keys`, `Object.create` en JSON
6. 2015: ES6/ECMAScript2015 komt uit (geen grote veranderingen, problemen om consensus te vinden over een aantal onderwerpen)

De JavaScript taal is niet beperkt tot browsers of internettoepassingen alleen. Je kan bijvoorbeeld de taal uitstekend gebruiken om binnen een Windows omgeving administratieve scripts te schrijven. De Windows Scripting Host is de vertolker die Microsoft in zijn besturingssysteem heeft geïntegreerd om dit te realiseren. Applicaties kunnen ook de mogelijkheid bieden om scripts uit te voeren om op die manier de applicatie zelf uit te breiden en flexibel te maken. Een voorbeeld is Flash dat ook JavaScript beheerst. Een softwareproducent die dergelijke mogelijkheden voor zijn applicatie wil aanbieden dient niet zelf JavaScript te implementeren, maar kan gebruik maken van de engines die door de Mozilla-organisatie in open source worden aangeboden.

Wanneer er gebruik gemaakt wordt van JavaScript is het niet de bedoeling om een server continu te connecteren om extra gegevens op te halen. Alle code moet dus voorradig zijn in de webpagina zelf. Het gebruik van JavaScript in een webbrowser wordt aangeduid als **client-side** JavaScript. Typisch surft een gebruiker immers met een PC of laptop (een client machine) op het internet. Eigenlijk is deze benaming niet zo goed gekozen, want je kan natuurlijk ook nog andere netwerkprogramma's op een client draaien. Men had beter gesproken over **browser-side** JavaScript, maar deze term bestaat niet. Denk er dus aan dat als je leest over client-side JavaScript, je eigenlijk de taal beschouwt zoals die in een webbrowser ingebed is.

De Core van de taal definieert enkele standaardobjecten (bv. het String object) die je als programmeur altijd en overal kan gebruiken. Aan de client-side bestaan er nog enkele andere objecten, die het mogelijk maken om vanuit JavaScript alle belangrijke onderdelen van een webapplicatie aan te spreken.

Tenslotte willen we nog even vermelden dat JavaScript helemaal niet gerelateerd is aan de taal Java. Beide zijn totaal verschillend qua historie en bedoeling. Het enige wat je kan zeggen is dat de syntax van beide talen erg gelijkend is, omdat ze allebei afstammen van C-taal.

12.2 De <script> tags en je eerste javascript

De <script>-tag wordt gebruikt om de client-side scripts toe te voegen aan je html document.

```
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="UTF-8">
  <title>Demo 1: first javascript</title>
</head>
<body>
  <output id="demo"></output>
  <script>
    document.getElementById("demo").innerHTML = "Hello JavaScript!";
  </script>
</body>
</html>
```

Het <script> element bevat de programma-logica, of kan verwijzen naar een extern bestand met het script.

```
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="UTF-8">
  <title>Demo 1: first javascript</title>
  <script src="info.js" async></script>
</head>
<body>
  <div>
    <output id="demo"></output>
  </div>
  <div>
    <output id="demo2"></output>
  </div>
  <script>
    document.getElementById("demo").innerHTML = "Hello JavaScript!";
  </script>
</body>
</html>
```

En het bestand info.js:

```
window.addEventListener("load", displayInfo);

function displayInfo() {
  document.getElementById("demo2").innerHTML = "I'm enjoying JavaScript";
}
```

Indien we de JavaScript code in een extern document schrijven plaatsen we de `<script>` tag binnen de `<head>` tag en gebruiken we `async` of `defer`. Dit omwille van de performantie van onze webpagina.

- Bij gebruik van **async**: Het script wordt asynchroon gedownload en uitgevoerd, dus tegelijkertijd met het verwerken van de rest van de html-pagina (het script wordt dus uitgevoerd terwijl de pagina verder wordt geparsed).
- Bij gebruik van **defer**: Het script wordt gedownload en uitgevoerd zodra de html-pagina verwerkt is (dus de browser klaar is met het parsen (verwerken) van de html-code).
- Zonder `defer` of `async`: Het script wordt direct opgehaald en uitgevoerd, daarna gaat de browser pas verder met het verwerken van de rest van de webpagina. Dit heeft dus grotere impact op de laadtijd van de webpagina. Daarom werd vroeger de script-tag pas aan het einde van de pagina geplaatst.

12.3 Lexicale structuur

Bij het schrijven van JavaScript moeten een aantal regels in het oog gehouden worden: Hoofdletters moeten gerespecteerd worden, statements correct gescheiden en commentaar op de juiste plaats gezet.

❖ Hoofdletters

Aangezien JavaScript hoofdlettergevoelig (case sensitive) is, dien je op te letten wanneer je variabelen of functies definieert met hoofdletters. Indien je deze variabelen of functies nadien wil gebruiken moet de schrijfwijze exact overeenkomen. Voor namen van variabelen en functies wordt vaak de camelCase notatie gehanteerd. In deze notatie wordt alles aan elkaar geschreven, maar beginnen alle woorden met een hoofdletter, uitgezonderd het eerste woord.

Ook bij gebruik van methodes, bijvoorbeeld `getElementById()`, moet de naam correct ingegeven worden.

```
var mijnVariabele = 0; //declareer een variabele

Mijnvariabele = 10; //Dit is fout
MijnVariabele = 10; //Dit is fout

mijnVariabele = 10; //Dit is correct
```

❖ Scheiden van statements

Statements (één enkele instructie) worden gescheiden door puntkomma's (;). Hoewel je ze soms kan weglaten, is het aangeraden om ze consistent en overal te schrijven om fouten te vermijden. Het is gemakkelijker ze steeds te plaatsen dan aan te leren wanneer ze niet nodig zijn.

Je kan meerdere statements op één lijn ingeven, maar tracht slechts één statement per lijn te gebruiken, dit verhoogt de leesbaarheid van je programma.

```
a = 10; b = 20;
```

of beter

```
a = 10;
b = 20;
```

❖ Commentaar

De volgende regel geeft aan hoe je commentaar binnen JavaScript kan gebruiken, dit is dezelfde methode als binnen CSS gebruikt wordt.

```
//deze functie toont een boodschap
```

Als je C++ of Java kent, dan ben je hier al mee vertrouwd. Twee schuine strepen (//) dienen voor commentaar op één lijn, terwijl meerdere lijnen commentaar worden afgebakend door de tekens /* en */.

```
/*
Dit zijn meerdere lijnen commentaar
Commentaar kan de code verduidelijken
*/
```

❖ Voorbehouden woorden

Er zijn enkele woorden die je beter niet gebruikt om variabelen en functies te benoemen. Dit omdat ze al in gebruik zijn in de JavaScript-taal zelf en omdat dit anders verwarring zou scheppen. Ter volledigheid volgt hier het lijstje.

Woorden gemarkeerd met een * zijn toegevoegd in ES (ECMAScript) 5 en 6.

abstract	arguments	await*	boolean
break	byte	case	catch
char	class*	const	continue
debugger	default	delete	do
double	else	enum*	eval
export*	extends*	false	final
finally	float	for	function
goto	if	implements	import*
in	instanceof	int	interface
let*	long	native	new
null	package	private	protected
public	return	short	static
super*	synchronized	switch	this
throw	throws	transient	true
try	typeof	var	void
volatile	while	with	yield

❖ Strict mode

De strict mode is toegevoegd sinds ES5 en geeft aan dat de code “streng” en “nauwkeurig” wordt uitgevoerd. We zorgen ervoor dat deze directive ALTIJD aan ons script wordt toegevoegd!

Met strenge uitvoering bedoelen o.a. dat:

- variabelen steeds gedeclareerd moeten worden
- hergebruik van parameter naam niet is toegestaan
- octaalgetallen en escape karkaters niet zijn toegestaan
- ...

```
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
  <output id="output"></output>
```



```

<script>
  "use strict";

  document.getElementById("output").innerHTML = double(5, 6);

  function double(value1, value1) {
    var oct = 010; // 1 * 8 = 8
    twice = 2;
    return value1 * twice * oct;
  }
</script>
</body>
</html>

```

Bovenstaand code-voorbeeld geeft bij niet-strikte uitvoering 96 als resultaat, bij strikte uitvoering krijg je dus verschillende foutmeldingen:

- Uncaught SyntaxError: Duplicate parameter name not allowed in this context
- Uncaught SyntaxError: Octal literals are not allowed in strict mode.
- Uncaught ReferenceError: twice is not defined

12.4 Variabelen

❖ Getallen

JavaScript gebruikt 64 bits om een nummer op te slaan. Je kan dus niet oneindig grote nummers maken.

```

var decGetal = 10;           // decimale waarde is 10
var hexGetal = 0x10;         // decimale waarde is 16
var octGetal = 0o10;         // decimale waarde is 8
var hexGetal = -0x10;        // decimale waarde is -16
var valversnelling = 9.81;   // komma-getallen
var grootGetal = 2.998e8;     // 2.998 × 108 = 299 800 000

```

❖ Wiskundig operatoren

Rekenkundige operatoren worden gebruikt voor wiskundige bewerkingen op numerieke waarden.

operator	bewerking	voorbeeld	commentaar
+	optellen	i=j+3;	
-	afrekken	i=j-3;	
*	vermenigvuldigen	i=j*3;	
/	delen	i=j/3;	
%	modulus	i=j%3;	rest van de deling
++	plus 1	i++;	idem aan i=i+1;
--	min 1	i--;	idem aan i=i-1;

❖ Toekenningoperatoren

Bij het toekennen van variabelen kunnen nog meerdere operatoren gebruikt worden. Buiten het gelijk-aan teken, kunnen al deze operatoren ook voluit geschreven worden met de rekenkundige operatoren, zoals in de laatste kolom getoond wordt.

operator	bewerking	voorbeeld	commentaar
----------	-----------	-----------	------------

=	gelijk aan	i=j	
+=	optellen met	i+=j	idem aan i=i+j
-=	afrekken met	i-=j	idem aan i=i-j
=	vermenigvuldigen met	i=j	idem aan i=i*j
/=	delen met	i/=j	idem aan i=i/j
%=	modulus met	i%=j	idem aan i=i%j

❖ Speciale getallen

Er zijn drie speciale waarden in JavaScript die we als getallen beschouwen, maar die zich niet gedragen als normale getallen.

De eerste twee zijn `Infinity` en `-Infinity`, die stellen dus positief oneindig en negatief oneindig voor. `Infinity - 1` geeft `Infinity`. De implementatie hiervan is niet volledig wiskundig onderbouwd en je kan er dus best niet te veel op vertrouwen als je oneindig-gebaseerde bewerkingen wil doen. Bewerkingen met `Infinity` en `-Infinity` leiden als snel tot het volgende speciale getal: `NaN`.

`NaN` staat voor “not a number”, ondanks het feit dat het een waarde voor een getal is. Je krijgt die als resultaat voor bijv. `0 / 0` (nul delen door nul) of `Infinity - Infinity`, ...

```
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="UTF-8">
  <title>Speciale getallen</title>
</head>
<body>
<output id="output"></output>
<script>
  var waarde1 = 120;
  var waarde2 = 0;
  var result1 = waarde1 / waarde2;
  var result2 = waarde1 / "abc";
  document.getElementById("output").innerText =
    `Deling door nul geeft: ${result1}.
    Deling door string geeft: ${result2}`;
</script>
</body>
</html>
```

We bekijken een concreet programma. Dit voorbeeld leest een getal in en rondt het af naar de dichtstbijzijnde gehele waarde.

```
function roundNumber() {
  var getal = prompt("Geef een getal in: ");
  if (isNaN(getal)) {
    return "U heeft geen getal ingegeven: " + getal;
  } else {
    var afrond = parseInt(getal + 0.5);
    return "Het getal " + getal + " is afgerond naar " + afrond;
  }
}
```

In bovenstaand programma wordt getest of een ingegeven waarde een nummer is of niet. Dit gebeurt met de functie `isNaN()` die `true` geeft indien de waarde geen getal is of `false` indien het wel een getal is.

```
if (isNaN(getal))
```

❖ String

Dit is een reeks van ASCII-karakters. Ze worden tussen enkele of dubbele aanhalingstekens geplaatst.

```
var welkomtekst = "Hello, World!";
var welkomtekst = 'Hello, Planet!';
```

```
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="UTF-8">
  <title>Haiku</title>
</head>
<body>
<output id="output"></output>
<script>
  var webSite = "Web site";
  var haiku = `The ${webSite} you seek
              cannot be located but
              endless others exist`;
  document.getElementById("output").innerText = haiku;
</script>
</body>
</html>
```

◆ Operatoren voor tekstvariabelen

Bij strings krijgt het plusteken een heel andere betekenis, namelijk de concatenatie of het samenvoegen van de tekst.

operator	bewerking	voorbeeld	commentaar
=	toekenning	woord=woord2	
+	concatenatie	woord=woord+woord2	idem aan woord += woord2
+=	concatenatie	woord+=woord2	idem aan woord=woord+woord2

Een klein voorbeeld maakt dit duidelijk.

```
var woord = "Hallo"
var woord2 = "Wereld"
woord += woord2           // woord bevat nu de string "HalloWereld"
```

◆ Property length

Deze (alleen lezen) property geeft aan uit hoeveel karakters de string bestaat.

```
var tekst = "Hogeschool PXL";
var lengte = tekst.length;           // lengte = 14
```

◆ String.charAt(pos)

Deze methode retourneert het karakter op positie `pos`. Let op: net zoals bij arrays begin je te tellen vanaf nul!

```
var tekst = "Hogeschool PXL";
var karakter = tekst.charAt(0);       // karakter = 'H'
```

◆ *String.indexOf(char, [start])*

Deze methode retourneert de eerste index waar het karakter of substring `char` binnen de string voorkomt. Je kan deze methode gebruiken om te zoeken binnen een string. Als het karakter of de substring niet gevonden wordt, retourneert de functie -1.

Optioneel kan een beginindex `start` meegegeven worden. Dit duidt aan vanaf welke positie er moet begonnen worden met zoeken.

```
var tekst = "Piet@myComp. be";
var i = tekst.indexOf('@');           // i = 4
var j = tekst.indexOf('. nl ');       // j = -1
```

◆ *String.substring(from, [to])*

Deze methode isoleert een substring uit de gegeven string. De eerste letter van de substring bevindt zich op index `from` en het argument `to` de positie van het eerste karakter dat *niet* meer tot de substring behoort. Als het wordt weggelaten, gaat het gewoon tot het einde van de string.

```
var tekst = "Hogeschool PXL";
var s1 = tekst.substring(0, 9);       // s1 = 'Hogeschool'
var s2 = tekst.substring(10);         // s2 = 'PXL'
```

◆ *String.toUpperCase()* en *String.toLowerCase()*

Converteert een string naar hoofdletters of naar kleine letters.

❖ boolean

Indien je slechts 2 mogelijke waarden nodig hebt zoals: 'ja' en 'nee', of 'aan' en 'uit', dan kan je het boolean type gebruiken. De waarden zijn dan **true** en **false**. Deze waarden zijn ook het resultaat bij het gebruik van een vergelijkende operator.

```
var aan = true;
```

```
var geslaagd = punten >= 10;
```

```
var test = "aaa" < "abc";
```

❖ Vergelijkende operatoren

Vergelijkende operatoren worden binnen controlestructuren (zie verder) toegepast en geven de waarde "waar" (true) of "niet waar" (false).

operator	bewerking	voorbeeld	commentaar
==	is gelijk aan	i==5	
===	waarde en type gelijk	i===5	dus enkel gelijk aan de integer 5, niet aan de string "5"
!=	is niet gelijk aan	i!=5	
!==	Waarde en type is niet gelijk	i !== 5	
>	is groter dan	i>5	
<	is kleiner dan	i<5	
>=	is groter of gelijk aan	i>=5	
<=	is kleiner of gelijk aan	i<=5	

JavaScript heeft 2 sets van operatoren om gelijkheid (of ongelijkheid) te testen: `===` en `!==`, en de slechtere varianten `==` en `!=`. De eerste (goede) werken zoals je zou verwachten: indien twee operanden hetzelfde type en waarde hebben, dan geeft `===` true en `!==` false als resultaat. De "evil twins" werken hetzelfde indien beide operanden van hetzelfde type zijn, maar indien ze een verschillend type hebben, zullen ze de waarden naar andere types dwingen volgens regels die erg ingewikkeld (en dus ook niet onthoudbaar) zijn. Hier enkele interessante voorbeeldjes:

```
' ' == '0'           // false
0 == ' '             // true
0 == '0'             // true

false == 'false'     // false
false == '0'         // true

false == undefined   // false
false == null        // false
null == undefined    // true

' \t\r\n ' == 0      // true
```

Het gebrek aan transitiviteit is een ernstig probleem. **Daarom: gebruik ALTIJD === en !==.**

❖ Logische operatoren

Dit zijn de Booleaanse operatoren and, or en not om een logische relatie tussen operandi te bepalen. Deze operatoren worden vaak in controlestructuren gebruikt. Bij de and-operator moeten beide vergelijkingen waar (true) zijn. Bij de or-operator moet één van beide operatoren waar (true) zijn, de andere mag zowel waar (true) of niet-waar (false) zijn. De not-operator mag vertaald worden als dit niet-waar (false) is.

operator	bewerking	voorbeeld	commentaar
&&	and	i == 0 && j < 5	controleert de en-relatie
	or	i == 0 i >= 100	controleert de of-relatie
!	not	!(i == j)	is eigenlijk een negatie en kan ook als i != j genoteerd worden

❖ null

Dit gereserveerde woord betekent zoveel als een lege variabele die dus geen inhoud of waarde heeft. Dit mag niet verward worden met de waarde van het getal nul. Door bij de declaratie de waarde null mee te geven maak je duidelijk dat dit om een object gaat, dat nog geen waarde heeft toegewezen gekregen. Bij stringvariabelen is er dus ook een duidelijk verschil met het toewijzen van een lege string met "" als waarde. Hiermee definieer je de string duidelijk als leeg, terwijl je met null aangeeft dat de stringwaarde nog aangemaakt moet worden.

❖ Undefined

Een variabele waaraan geen waarde is toegekend heeft type undefined. Je kan op onderstaande manieren testen of een variabele x undefined is:

```
var x;
if (typeof x === 'undefined') {
    // these statements execute
}
OF
if (x === undefined) {
    // these statements execute
}
```

Nog een voorbeeldje met berekeningen met null en undefined. Een berekening met undefined geeft NaN, bij berekeningen met null wordt null naar 0 geconverteerd.

```
<!DOCTYPE html>
<html lang="nl">
<head>
```

```

<meta charset="UTF-8">
<title>Null en undefined</title>
</head>
<body>
<output id="outputCirkel1"></output><br/>
<output id="outputCirkel2"></output>
<script>
    var straalCirkel1;
    var omtrekCirkel1 = straalCirkel1 * 2 * Math.PI;
    document.getElementById("outputCirkel1").innerHTML =
        "De omtrek van een cirkel met straal "
        + straalCirkel1 + " is " + omtrekCirkel1;
    var straalCirkel2 = null;
    var omtrekCirkel2 = straalCirkel2 * 2 * Math.PI;
    document.getElementById("outputCirkel2").innerHTML =
        "De omtrek van een cirkel met straal "
        + straalCirkel2 + " is " + omtrekCirkel2;
</script>
</body>
</html>

```

❖ Zwakke typering

Het type van de variabele is niet opgegeven, maar wordt door de uitvoeringsomgeving afgeleid uit de context. JavaScript is dus een **zwak getypeerde** scripttaal in tegenstelling tot bijvoorbeeld C en Java wat sterk getypeerde programmeertalen zijn. In deze talen dient een typedeclaratie te gebeuren en wordt het type meegegeven, bijvoorbeeld `int` voor integers. In onderstaand C-voorbeeld declareer je een variabele `i` van het type `integer`.

```

/* C voorbeeld: gebruik van variabelen */
int i = 10;      // type: integer
i = i * 5;      // gebruik integers enkel volgens de regels

```

In JavaScript is dit niet nodig en wordt altijd `var` gebruikt (of `let` of `const`). Je moet variabelen dus nog altijd declareren (vandaar het woord `var`), maar hun type wordt automatisch door de vertolker bepaald uit de context, dit noemt men **automatische typeconversie**.

```

/* JavaScript voorbeeld: automatische types */
var i = 10;      // type is duidelijk uit context
i = "Dit is een getal: " + i; // conversie

```

In het voorbeeld ken je eerst het getal 10 toe aan de variabele `i`. Op dat moment is `i` van het type `integer`. Bij het optellen van `i` bij een string, zal `i` omgezet worden naar een string en zal het optelteken geïnterpreteerd worden als een concatenatie. Het eindresultaat is dus dat `i` een stringvariabele is.

❖ Bereik van een variabele

Het bereik (scope) bepaalt waar en wanneer een bepaalde variabele beschikbaar is. Het eenvoudigst zijn **globale variabelen**: zij worden gedeclareerd buiten de functies maar binnen de `script`-tag en zijn bijgevolg voor alle functies toegankelijk. Binnen de functies kunnen ook **lokale variabelen** gedeclareerd worden die dan ook enkel beschikbaar zijn binnen deze functie. Het declareren van een variabele is overal mogelijk binnen de functie.

```

<!DOCTYPE html>
<html lang="nl">
<head>
    <meta charset="UTF-8">
    <title>Globale en lokale variabelen</title>
</head>
<body>

```

```

<output id="output"></output>
<script>
    var getal1 = 5; // globale variabelen
    var getal2 = 10;

    function vermeningvuldig() {
        var getal2 = 7; // lokale variabele
        document.getElementById("output").innerHTML = getal1 * getal2;
    }

    vermeningvuldig()
</script>
</body>
</html>

```

❖ Let en const (ES6)

De **const declaratie** wordt gebruikt voor een read-only variabele, een constante. Wanneer aan een constante een waarde is toegekend, kan je er geen andere aan toewijzen.

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Const</title>
</head>
<body>
<output id="output"></output>
<script>
    "use strict";
    const MINUTES_IN_HOUR = 60;

    MINUTES_IN_HOUR = 40; //Uncaught TypeError: Assignment to constant variable.

    function aantalMinuten(aantalUren) {
        return hour * MINUTES_IN_HOUR;
    }

    document.getElementById("output").innerHTML =
        "3 uur = " + aantalMinuten(3) + " min";
</script>
</body>
</html>

```

De **let declaratie** wordt gebruikt voor variabelen waarvan de waarde wel mag veranderen tijdens de uitvoering van het programma (zoals een teller in een herhaling, ...). Verder zorgt let er ook voor dat de variabele enkel gebruikt kan worden in het code-blok waarin de variabele wordt gedefinieerd. Dit laatste is niet het geval bij gebruik van var zoals onderstaand voorbeeld laat zien.

```

<!DOCTYPE html>
<html lang="nl">
<head>
    <meta charset="UTF-8">
    <title>Var en let</title>
</head>
<body>
<output id="output"></output>
<script>
    "use strict";

    function optellen() {

```

```

    let resultaat = "";
    for (let teller = 0; teller <= 5; teller++) {
        resultaat += teller + "<br/>";
    }
    for (var andereTeller = 6; andereTeller <= 10; andereTeller++) {
        resultaat += andereTeller + "<br/>";
    }
    resultaat += "teller=" + teller + "<br/>"; // ReferenceError
    resultaat += "andereTeller=" + andereTeller + "<br/>";
    return resultaat;
}
document.getElementById("output").innerHTML = optellen();
</script>
</body>
</html>

```

Je krijgt nu een foutmelding “Uncaught ReferenceError: teller is not defined”, terwijl andereTeller wel gekend is na de for-lus (en de waarde 11 heeft).

Gebruik dus steeds let bij for-lussen!

In strict mode zal het bij gebruik van let ook niet mogelijk zijn een variabele met dezelfde naam te herdeclaren.

```

'use strict';
let me = 'foo';
let me = 'bar'; // SyntaxError: Identifier 'me' has already been declared

var me = 'foo';
var me = 'bar'; // No problem, `me` is replaced.

```

❖ Arrays

Een variabele kan telkens maar één waarde bevatten, maar soms is het nuttig om bepaalde variabelen te koppelen in een reeks van gerelateerde data. Net zoals in de programmeertaal Java (en andere programmeertalen) kan hiervoor een array of reeksvariabele gebruikt worden. De reeksvariabele is dus een rij van objecten, maar wordt opnieuw voorgesteld door een object. Het declareren van een array kan op verschillende manieren.

```

var leeg = new Array(); // geen bepaalde grootte, leeg
var weekdag = new Array(7); // grootte 7, leeg
var cijfers = new Array(10); // grootte 10, leeg
var cijfers = new Array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9); // grootte 10, bepaald

```

De eerste reeks is leeg, de tweede reeks weekdag heeft plaats voor 7 elementen, maar die zijn nog niet gedefinieerd. De derde reeks cijfers heeft 10 ongedefinieerde elementen, terwijl deze in de vierde reeks wel bepaald zijn. De ingevulde variabelen hoeven niet noodzakelijk van hetzelfde type te zijn.

Het benaderen van de elementen van de reeks gebeurt met vierkante haakjes. Let op: de indexen beginnen te tellen vanaf nul!

```

var maand = new Array(12);
maand[0] = "januari";
maand[1] = "februari";
...
maand[11] = "december";

```

Dit kan ook korter in gecondenseerde opmaak:

```
var maand = new Array("j anuari", "februari", ..., "december");
```

Of in letterlijke opmaak, gebruik makend van vierkante haken.

```
var maand = new Array();
var maand = ["j anuari", "februari", ..., "december"];
```

Het aantal elementen in een array wordt bijgehouden in de property `length`.

```
var n = maand.length;
maand[n-1] = "december";           // laatste element
```

Let erop dat je de juiste indexen gebruikt, want als je een foutieve (te grote) index gebruikt, wordt de reeks gewoon stilzwijgend groter gemaakt. Hierin verschilt JavaScript van Java waar er een `ArrayIndexOutOfBoundsException` zou worden teruggegeven.

```
maand[12] = "Leve de dertiende maand!"; // maand bevat nu 13 elementen
```

12.5 Controlestructuren

Controlestructuren zorgen ervoor dat er in een programma beslissingen genomen worden en berekeningen uitgevoerd. Deze constructies komen min of meer letterlijk voor in de meeste programmeertalen. We bespreken ze daarom bondig.

❖ if-else

Deze structuur laat toe om op basis van een voorwaarde een beslissing te nemen. De voorwaarde staat tussen haakjes en maakt gebruik van een vergelijkingsoperator of een boolean. Ze geeft als resultaat een booleaanse waarde (`true` of `false`) terug. Indien er meerdere statements uitgevoerd moeten worden na de `if` of `else`, dan groepeer je de statements met accolades.

```
if (voorwaarde) {
    statements;
}
```

Bijvoorbeeld:

```
if (leeftijd >= 18) {
    document.getElementById("output").innerHTML = "+ 18";
}
```

In dit voorbeeld wordt getest of de variabele `leeftijd` groter is dan 18.

Om te controleren of een variabele een geldige waarde heeft kan je de volgende voorwaarde gebruiken:

```
if( value ) {
}
```

Dit wordt geëvalueerd naar `true` indien `value` niet een van de volgende waarde is:

- `null`
- `undefined`
- `NaN`
- `empty string ("")`
- `0`
- `false`

Je kan meerdere voorwaarden samenvoegen door de operatoren `&&` en `||`. Negatie van een test doe je met `!` (zie logische operatoren). Let erop dat je de haakjes correct gebruikt!

De `if`-structuur kan uitgebreid worden met een `else`-gedeelte.

```
if (voornaam && achternaam) {
    document.getElementById("output").innerHTML =
        "Hallo, " + voornaam + " " + achternaam;
}
else {
    document.getElementById("output").innerHTML = "Hallo, John Doe!";
}
```

De `if`-structuur kan verkort weergegeven worden door middel van de voorwaardelijke operator, die geschreven wordt als een vraagteken (`?`).

```
var teken = (i >= 0)? 'positief': 'negatief';
```

In bovenstaand voorbeeld zal de stringvariabele dus het woord `positief` bevatten als `i` groter of gelijk is aan nul, in het andere geval bevat ze het woord `negatief`.

❖ switch

De `switch`-constructie is een veralgemening van de `if-else` structuur en laat toe om op een overzichtelijke manier verschillende mogelijke beslissingen te evalueren. Dit is veel leesbaarder en handelbaarder dan een geneste `if-else` structuur.

In het voorbeeld is de uitvoering van het programma afhankelijk van de waarde van de variabele `n`. Als `n` gelijk is aan 1 wordt het bijhorende stuk code dat volgt op `case 1` uitgevoerd. Bemerk het `break`-statement dat ervoor zorgt dat er niet verder gegaan wordt met `case 2`, maar dat er correct uit de `switch` wordt gesprongen. Een typische fout is het vergeten van het `break`-statement. Tenslotte staat een `default`-gedeelte ter beschikking voor alle andere gevallen.

```
switch(n) {
    case 1:                //voer code uit horende bij n == 1
        statement1;
        break;            //spring uit het switch statement
    case 2:                //voer code uit horende bij n == 2
        statement2;
        break;            //spring uit het switch statement
    case 3:                //voer code uit horende bij n == 3
        statement3;
        break;            //spring uit het switch statement
    default:               //voer code uit horende bij alle andere gevallen
        statementn;       //spring uit het switch statement
}
```

❖ while

Een lus kan gemaakt worden met behulp van een `while`-statement.

```
while (voorwaarde) {
    statement(s);
}
```

Zolang aan de voorwaarde voldaan wordt (`true`), worden de statements uitgevoerd. Het volgende voorbeeld drukt de even getallen kleiner dan 10 af.

```
function printEven() {
    var tekst = "Even getallen: ";
    var i = 0;
```

```

while (i < 10) {
    if (i % 2 === 0) {
        tekst += i + " ";
    }
    i++;
}
document.getElementById("output").innerHTML = tekst;
}

```

❖ do-while

De `do-while`-constructie is een alternatief voor de `while`-constructie. Het verschil is dat de statements altijd minstens éénmaal worden uitgevoerd.

```

do {
    statement(s);
} while (voorwaarde);

```

De statements worden uitgevoerd zolang de expressie waar is. Let ook op de puntkomma (;) na de voorwaarde. Hier volgt de aanpassing om de even getallen te printen.

```

function printEven () {
    var tekst = "Even getallen: ";
    var i = 0;
    do {
        if (i % 2 === 0) {
            tekst += i + " ";
        }
        i++;
    } while (i < 10);
    document.getElementById("output").innerHTML = tekst;
}

```

❖ for

De `for`-lus maakt het werken met tellers wat eenvoudiger en is de aangewezen constructie wanneer het aantal herhalingen van tevoren gekend is. Tussen de haakjes komt eerst de initialisatie waarbij de teller op een waarde ingesteld wordt, daarna volgt de voorwaarde. Zolang er aan de voorwaarde voldaan wordt en dus de waarde `true` terugkomt, blijft het programma in de `for`-lus, anders (`false`) springt ze eruit. Als laatste komt de definitie van de stappen, meestal een ophoging in de vorm van `i++`.

```

for (initialisatie; voorwaarde; stappen) {
    statement(s);
}

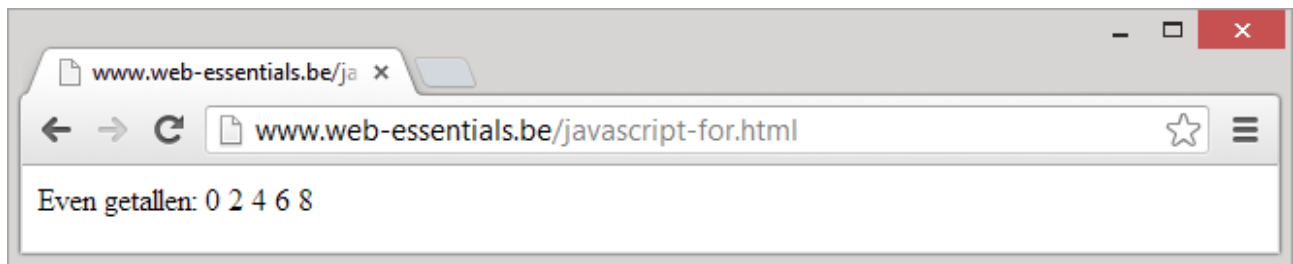
```

We herhalen het voorbeeld met de even getallen.

```

function printEven () {
    var tekst = "Even getallen: ";
    for (let i = 0; i < 10; i++) {
        if (i % 2 === 0) {
            tekst += i + " ";
        }
    }
    document.write(tekst);
}

```



12.6 Functies

❖ Functie expressie en functie declaratie

Er zijn twee manieren om functies te definiëren in JavaScript. De eerste manier is de functie expressie.

Een **functie expressie** is eigenlijk een gewone variabele declaratie waarbij de waarde van de variabele een functie is. In het onderstaande voorbeeld hebben we een variabele *square* waaraan een functie (die een kwadraat berekent) is toegekend.

```
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="UTF-8">
  <title>Demo9: functie</title>
</head>
<body>
  <output id="square"></output>
  <output id="cube"></output>
  <script>
    "use strict";
    var getal1 = 5;
    var getal2 = 4;

    document.getElementById("square").innerHTML = square(getal1);
    // bovenstaande regel geeft: "Uncaught TypeError: square is not a function"
    document.getElementById("cube").innerHTML = cube(getal1);

    var square = function(x) {
      return x * x;
    };

    function cube(x) {
      return x * x * x;
    }

    document.getElementById("square").innerHTML = square(getal2);
    document.getElementById("cube").innerHTML = cube(getal2);
  </script>
</body>
</html>
```

Voor de functie *cube* gebruiken we een **functie declaratie**. Deze vorm van definiëren van functies heeft één groot verschil met de **functie expressie**: de functie declaratie geeft je meer vrijheid over de plaats waar je ze plaats, omdat de functie ook beschikbaar is in code die voorafgaat aan de functie declaratie.

❖ Geneste functies

```

<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="UTF-8">
  <title>Demo10: nested functions</title>
</head>
<body>
<output id="output"></output>
<script>
  var landscape = function() {
    var result = "";
    var flat = function(size) {
      for (let count = 0; count < size; count++)
        result += "_";
    };
    var mountain = function(size) {
      result += "/";
      for (let count = 0; count < size; count++)
        result += "'";
      result += "\\\";
    };

    flat(3);
    mountain(4);
    flat(6);
    mountain(1);
    flat(1);
    return result;
  };

  document.getElementById("output").innerHTML = landscape();
</script>
</body>
</html>

```

❖ Immediately-Invoked Function Expression (IIFE)

IIFE (uitspraak: "Ifly") is de afkorting voor Immediately-Invoked Function Expression, en de syntax ziet er als volgt uit:

```

(function () {
  // do fun stuff
})();

of

(function () {
  // do fun stuff
})();

```

Het is een anonieme functie die onmiddellijk wordt uitgevoerd. Deze constructie wordt gebruikt om te vermijden dat variabelen in de global scope gedeclareerd worden. Globale variabelen kunnen soms vreemde neven-effecten veroorzaken in je script (zeker bij concurrency (=het gelijktijdig uitvoeren van bepaalde stukken code)). Het is daarom aangeraden om je volledige script als een IIFY te schrijven.

```

<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="UTF-8">
  <title>Demo 11: IIFE</title>

```

```

<style>
  output {
    display: block;
  }
</style>
</head>
<body>
<output id="output1"></output>
<output id="output2"></output>
<output id="output3"></output>
<script>
  (function () {
    (function () {
      var now = new Date();
      var start = new Date(now.getFullYear(), 0, 0);
      var diff = now - start;
      var oneDay = 1000 * 60 * 60 * 24;
      var days = Math.floor(diff / oneDay);
      document.getElementById("output1").innerHTML =
        "Dag van het jaar:" + days;
    })();

    (function (bedrag) {
      var incBtw = 1.21;
      document.getElementById("output2").innerHTML =
        "Bedrag inc. btw: " + (incBtw * bedrag);
    })(120);

    var min = Math.min(15, (function () {
      return new Date().getDate();
    })());

    document.getElementById("output3").innerHTML = min;
  })();
</script>
</body>
</html>

```

❖ Functie met return-waarde functie

```

<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="UTF-8">
  <title>Demo12: functie return functie</title>
</head>
<body>
<output id="output"></output>
<script>
  (function() {
    var btw = function (procent) {
      return function (bedrag) {
        return bedrag * (1 + procent / 100);
      };
    };

    var btwEenEnTwintig = btw(21);
    var btwZes = btw(6);

    document.getElementById("output").innerHTML =
      btwEenEnTwintig(120) + " " + btwZes(120);
  })();
</script>

```

```
</body>
</html>
```

12.7 Objecten

Objectgeoriënteerd programmeren wordt door veel talen ondersteund, ook door JavaScript. In dit deel zie je kort hoe JavaScript objecten aanbiedt en hoe je als programmeur ermee overweg kan.

❖ Wat is een Object?

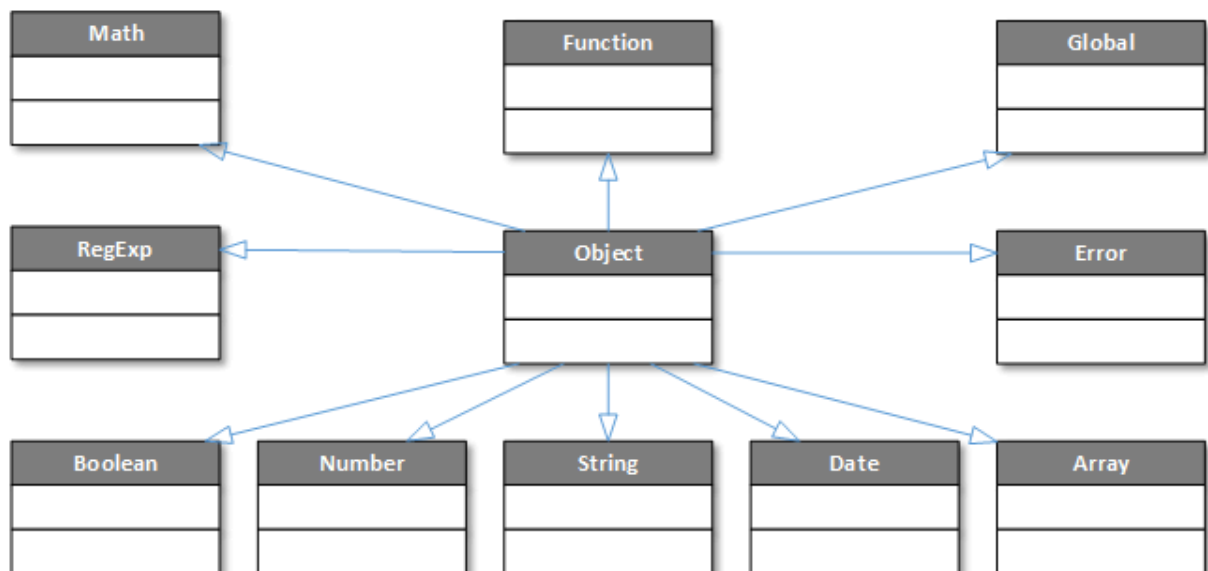
Het is niet de bedoeling om hier volledig uit te leggen wat een object is. Daarover bestaan er meerdere handboeken. In een programma bestaat een **object** uit een set van waarden (**properties**) en **methods**. Een method kan de toestand (property) van een object wijzigen.

Een `String`, een `Array`, een `Window`, een `Frame`, een `Form`, ... zijn allemaal objecten die eigenschappen bevatten en zo een **entiteit** vormen.

Een object heeft **properties** of eigenschappen die je kunt lezen en/of schrijven. Bijvoorbeeld een `window` heeft een `name` die je kunt zetten of opvragen. De property van een object kan opnieuw objecten opleveren, bijvoorbeeld een `window` heeft een property `frames[]` die een array van frame-objecten oplevert.

```
object.property
object.method()
```

Een object heeft **methodes** die je kan oproepen. Deze kunnen de toestand van dat object wijzigen. Bijvoorbeeld een `window` heeft een methode `close()` om dit venster te sluiten. Onderstaande figuur geeft een overzicht van de objecten die JavaScript ter beschikking stelt aan de programmeur. Je kan ze dus gebruiken in eender welke omgeving die de JavaScript standaard ondersteunt.



◆ *Object*

Dit is de basisklasse. Elk object binnen JavaScript erft de properties en methods van dit object over. Bijvoorbeeld de methode `Object.toString()` geeft een string voorstelling van het betreffende object.

◆ *Boolean*

Deze objecten fungeren als omhulsel (eng: wrapper) voor de booleaanse waarden `true` en `false`.

◆ **Number**

Objecten van het type `Number` stellen getallen voor. Deze klasse heeft enkele belangrijke properties, bijvoorbeeld `Number.NaN`, `Number.MAX_VALUE` en `Number.MIN_VALUE`.

◆ **String**

Een `String` is een karakters voorstelling van tekstuele informatie.

◆ **Date**

Objecten die dienen om te werken en te rekenen met datums.

◆ **Array**

Dit is een datastructuur die bestaat uit een lijst van elementen (objecten).

◆ **RegExp**

De klasse `String` biedt reeds heel wat methodes die helpen met het zoeken van woorden en patronen. Deze klasse gaat hierop verder en biedt de programmeur *reguliere expressies*. Oorspronkelijk afkomstig uit de Unix-wereld en de taal Perl, is het hier mogelijk om ingewikkelde zoekpatronen op te bouwen.

◆ **Math**

Dit een bibliotheek van de meest gebruikte wiskundige functies en constanten.

◆ **Global**

Een verzameling van alle globale objecten. Als je zelf een globale variabele declareert erft die de properties en methods over van de klasse `Global`.

◆ **Error**

Net zoals Java en C++, biedt JavaScript een exception mechanisme. We gaan hier niet verder op in, maar vermelden dat de exceptions die gegooid worden van het type `Error` zijn.

❖ **Objecten maken**

```
var vandaag = Date.now();
var kerstmis = new Date(2009, 12, 25);
```

Meestal hoef je geen objecten aan te maken, maar gebruik je reeds beschikbare objecten. Bijvoorbeeld binnen een browser omgeving beschik je over de objecten `document` en `window`.

❖ **Properties en methodes**

Properties benader je door middel van de zogenaamde *dot-notatie*. Aan de linkerkant van de punt is een referentie naar het object, de rechterkant is de beschouwde property. Een methode is een functie die hoort bij een object. Ze wordt net zoals properties opgeroepen door middel van de *dot-notatie*. Een voorbeeld maakt veel duidelijk.

```
var myField = document.getElementById("field"); // methode
myField.style.backgroundColor = "white"; // properties
```

13 Document Object Model

13.1 Browser Object Model (BOM)

Het **window**-object wordt door alle browsers ondersteund en is een referentie naar het browserscherm. Alle globale variabelen en functies zijn beschikbaar op het window-object. Globale variabelen zijn eigenschappen van het window-object. Globale functies zijn methoden van het window-object.

```
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="UTF-8">
  <title>Demo: BOM</title>
  <script>
    var x = 120;

    var isEven = function(waarde) {
      return waarde && waarde % 2 == 0;
    }
  </script>
</head>
<body>
<output id="output" />

<script>
  var info = "x=" + window.x + " of " + x + "\n";
  info += "isEven(window.x)=" + window.isEven(x) + "\n";
  info += "afmetingen browserscherm (px)= " + window.innerHeight + " X " +
window.innerWidth + "\n";
  info += "url=" + window.location.href + "\n";
  info += "os=" + navigator.platform + "\n";

  document.getElementById("output").innerText = info;
</script>
</body>
</html>
```

13.2 Document Object Model (DOM)

Wanneer je een webpagina opent in de browser, dan haalt de browser de inhoud van de pagina op van de server en verwerkt vervolgens de inhoud van de pagina (dit noemen we “parsen”). Tijdens het parsen van de pagina bouwt de browser een model van de structuur van de webpagina. Dit model wordt dan gebruikt om de pagina op het scherm (of een ander medium) weer te geven.

Dit model, de voorstelling van de webpagina, kan je ook manipuleren in een Javascript programma. Je kan gegevens uit het model ophalen en aanpassen. Het model is een “levende” datastructuur: als je het model aanpast, zal de voorstelling van de webpagina op het scherm aangepast worden en je aanpassing dus zichtbaar zijn.

13.3 W3C DOM

Hier is een html-document:

```
<!DOCTYPE html>
<html lang="nl">
<head>
```



```

<meta charset="UTF-8">
<title>Demo 14: DOM</title>
</head>
<body>
<form id="myForm"
  action="mailto:nele.custers@pxl.be?subject=myAnswer"
  enctype="text/plain" method="post">
  <label for="naam">Naam:</label>
  <input type="text" id="naam" name="naam" />
  <fieldset id="vraag1">
    <legend>Inside which HTML element do we put the JavaScript?</legend>
    <input type="radio" name="scriptTag" value="js" id="js"/>
    <label for="js">&lt;js&gt;</label>
    <input type="radio" name="scriptTag" value="script" id="script"/>
    <label for="script">&lt;script&gt;</label>
    <input type="radio" name="scriptTag" value="javascript"
      id="javascript"/>
    <label for="javascript">&lt;javascript&gt;</label>
    <input type="radio" name="scriptTag" value="code" id="code"/>
    <label for="code">&lt;code&gt;</label>
  </fieldset>
</form>
<ul class="hinder">
  <li>Weet je het antwoord al?</li>
</ul>
</body>
</html>

```

De bijhorende DOM ziet er als volgt uit (gegenereerd met <http://software.hixie.ch/utilities/js/live-dom-viewer/>):

```

└─ DOCTYPE: html
└─ HTML lang="en"
  └─ HEAD
    └─ #text:
    └─ META charset="UTF-8"
    └─ #text:
    └─ TITLE
      └─ #text: Demo 14: DOM
    └─ #text:
  └─ #text:
  └─ BODY
    └─ #text:
    └─ FORM id="myForm" action="mailto:nele.custers@pxl.be?subject=myAnswer" enctype="text/plain" method="post"
      └─ #text:
      └─ LABEL for="naam"
        └─ #text: Naam:
      └─ INPUT name="naam" id="naam" type="text"
      └─ #text:
      └─ FIELDSET id="vraag1"
        └─ #text:
        └─ LEGEND
          └─ #text: Inside which HTML element do we put the JavaScript?
        └─ #text:
        └─ INPUT name="scriptTag" id="js" type="radio" value="js"
        └─ LABEL for="js"
          └─ #text: <js>
        └─ #text:
        └─ INPUT name="scriptTag" id="script" type="radio" value="script"
        └─ LABEL for="script"
          └─ #text: <script>
        └─ #text:
        └─ INPUT name="scriptTag" id="javascript" type="radio" value="javascript"
        └─ LABEL for="javascript"
          └─ #text: <javascript>
        └─ #text:
        └─ INPUT name="scriptTag" id="code" type="radio" value="code"
        └─ LABEL for="code"
          └─ #text: <code>
        └─ #text:
      └─ #text:
      └─ UL class="hinder"
        └─ #text:
        └─ LI
          └─ #text: Weet je het antwoord al?
        └─ #text:
      └─ #text:

```

De gebruikte methode om een door een `id`-attribuut benoemd element te benaderen is

```
var tekst = document.getElementById('naam').value;
```

Twee andere methodes zijn `getElementsByTagName()` en `getElementsByName()`. Let erop dat er hier in veelvoud (`Elements`) gesproken wordt aangezien er natuurlijk meerdere objecten met dezelfde naam of tagnaam kunnen zijn. Alle HTML-elementen die voldoen aan deze naam komen in een array terecht. Bij `getElementById()` is dit niet zo aangezien een `id`-attribuut uniek moet zijn.

```
var tekst = document.getElementsByTagName('input').item(0).value;
```

Indien er aan het `input`-element naast een `id` ook een `name`-attribuut was meegegeven, kan ook de volgende code.

```
var tekst = document.getElementsByName('naam').item(0).value;
```

of

```
var tekst = document.getElementById('naam')[0].value;
```

Zoals in de bovenstaande voorbeeldcodes is weergegeven kan van elk geadresseerd element een attribuutwaarde opgevraagd worden, door de naam van het attribuut mee te geven.

```
var titel = document.getElementById('identificatie').title;
var waarde = document.getElementById('identificatie').value;
var klasse = document.getElementById('identificatie').class;
var bron = document.getElementById('identificatie').src;
```

Bij het benaderen van een stijlregel van een HTML-element moet de term `style` opgegeven worden, gevolgd door de benaming van de stijldeclaratie. Indien deze benaming uit meerdere woorden bestaat dient ze aan elkaar vast geschreven te worden, waarbij het tweede woord met een hoofdletter begint.

```
var hoogte = document.getElementById('naam').style.height;
var kleur = document.getElementById('naam').style.color;
var achtergrond = document.getElementById('naam').style.backgroundColor;
var bovenmarge = document.getElementById('naam').style.marginTop;
```

Binnen het W3C-DOM kunnen deze methods verder aangevuld worden met de volgende formuleringen:

- `childNodes[]`, een reeks van alle knooppunten onder dat bepaalde element, sequentieel genummerd in de volgorde waarin ze in de HTML-code voorkomen.
- `firstChild`, het eerste onderliggend HTML-element.
- `lastChild`, het laatste onderliggend HTML-element.

Om met behulp hiervan terug de waarde van het `input`-element met de `id` naam op te vragen, kan dit met de volgende code. Ondanks het feit dat er maar één `body`-element mag aanwezig zijn in het `document`, moet hier toch het volgordenummer binnen de reeks meegegeven worden.

```
var tekst =
document.getElementsByTagName('body')[0].firstChild.firstChild.value;
```

of

```
var tekst =
document.getElementsByTagName('body')[0].childNodes[0].childNodes[0].value;
```

Vanzelfsprekend kan ook een mengvorm van beide methoden.

```
var tekst =
document.getElementsByTagName('body')[0].firstChild.childNodes[0].value;
```

of

```
var tekst =
document.getElementsByTagName('body')[0].childNodes[0].firstChild.value;
```

Nieuw vanaf HTML5 zijn de functies `document.getElementsByClassName()`, `document.querySelector()` en `document.querySelectorAll()`.

Met `document.getElementsByClassName()` kan je alle elementen kiezen waaraan je een bepaalde `class` hebt toegekend. Dit kan handig zijn om de stijl of inhoud van dergelijke elementen te veranderen.

```
<script>
```

```

function maakOranje () {
    var groep = document.getElementsByClassName('geel');
    for (let i = 0; i < groep.length; i++) {
        groep[i].style.color='orange';
    }
}
</script>

```

Met `document.querySelector()` kunnen CSS-selectors worden meegegeven om een element te definiëren. Enkel de eerste hit wordt opgenomen.

```

<script>
function maakOranje () {
    document.querySelector('p + p').style.color='orange';
}
</script>

```

Indien alle elementen die matchen met de opgegeven selector moeten geselecteerd worden in een reeks, dan dien je `document.querySelectorAll()` te gebruiken.

```

<script>
function maakOranje () {
    var groep = document.querySelectorAll('p + p');
    for (var i = 0; i < groep.length; i++) {
        groep[i].style.color='orange';
    }
}
</script>

```

We gebruiken nu de bovenstaande methodes om de DOM van onze pagina te gaan manipuleren:

```

<!DOCTYPE html>
<html lang="nl">
<head>
    <meta charset="UTF-8">
    <title>Demo 14: DOM</title>
</head>
<body>
<form id="myForm"
    action="mailto:nele.custers@pxl.be?subject=myAnswer"
    enctype="text/plain" method="post">
    <label for="naam">Naam:</label>
    <input type="text" id="naam" name="naam" />
    <fieldset id="vraag1">
        <legend>Inside which HTML element do we put the JavaScript?</legend>
        <input type="radio" name="scriptTag" value="js" id="js"/>
        <label for="js">&lt;js&gt;</label>
        <input type="radio" name="scriptTag" value="script" id="script"/>
        <label for="script">&lt;script&gt;</label>
        <input type="radio" name="scriptTag" value="javascript"
            id="javascript"/>
        <label for="javascript">&lt;javascript&gt;</label>
        <input type="radio" name="scriptTag" value="code" id="code"/>
        <label for="code">&lt;code&gt;</label>
    </fieldset>
</form>
<ul class="hinder">
    <li>Weet je het antwoord al?</li>
</ul>
<script>

    (function() {
        var scriptTags = document.getElementsByName("scriptTag");
        for (let counter = 0; counter < scriptTags.length; counter++){

```

```
        scriptTags.item(counter).value = "code";
    }
})();

var scriptTagsVerdwijen = function() {
    var scriptTags = document.getElementsByTagName("scriptTag");
    for (let counter = 0; counter < scriptTags.length; counter++){
        scriptTags.item(counter).style.visibility = "hidden";
    }
}

var submitForm = function() {
    document.getElementById("myForm").submit();
}

var hinder = function() {
    var newElement = document.createElement("li");
    var hinderLijst = document.getElementsByClassName("hinder")[0];
    hinderLijst.appendChild(newElement);
    hinderLijst.lastChild.innerText = "Weet je het al?";
}

setTimeout(scriptTagsVerdwijen, 9000);
setTimeout(submitForm, 6000);
setInterval(hinder, 1000);
</script>
</body>
</html>
```

14 DOM scripting

14.1 Inleiding: Wat is DOM-scripting?

Om browser- en platformonafhankelijk te werken is het noodzakelijk om enkel met algemeen erkende technologieën te werken, nl. **CSS**, **Javascript** en het **W3C DOM** (Document Object Model). Al deze standaarden worden door de huidige generatie browsers ondersteund.

Voor deze browseronafhankelijke combinatie wordt vaak de term DOM-scripting gebruikt. Er wordt gebruik gemaakt van het W3C DOM om van bepaalde HTML-elementen het uitzicht te wijzigen. Veelgebruikte toepassingen zijn onder andere:

- het zichtbaar en onzichtbaar maken van elementen of lagen door bijvoorbeeld de CSS-declaratie *visibility:visible* te wijzigen naar *visibility:hidden*.
- De kleur van bepaalde elementen wijzigen.
- De volledige opmaak van een pagina wijzigen.
- Het creëren van een roll-over afbeelding die dus wijzigt wanneer de muis erover gaat.
- Het programmeren van een diavoorstelling.
- Het valideren van dynamische formulieren.

Al deze functies worden in JavaScript geprogrammeerd en opgeroepen door een bepaalde trigger.

14.2 Event handling

Het gebruik van JavaScript binnen een webbrowser is hoofdzakelijk eventbased. De gebruiker leest de webpagina en reageert erop: formulieren verzenden, op links klikken, scrollen, animaties starten, ...

De gebeurtenis die de gebruiker initieert noemt men een **event**. Nu kan je aan deze gebeurtenis gehoor geven door een bepaald stuk programmacode uit te voeren. Dit noemt men **event handling**.

❖ Event handling via JavaScript

```
<type type="button" id="knop">...</button>
```

We kunnen op twee manieren events toevoegen aan html-elementen. De oudste manier voegt een event als methode toe, waarbij je dit aan een functie koppelt.

```
document.getElementById("knop").onclick = doeIets;

function doeIets() {
  ""
}
```

Bij de nieuwere gebruik je de methode "addEventListener". Onze voorkeur gaat uit naar deze nieuwere methode.

```
window.addEventListener('load', function() {
  document.getElementById("knop").addEventListener('click', doeIets);
});

function doeIets() {
  ""
}
```

❖ Event handlers voor documenten en vensters

◆ *onload*

Dit event treedt op nadat een object is geladen. Dit wordt zeer veel gebruikt bij het laden van een document en is zelfs onmisbaar als nieuwe pagina's dynamisch dienen opgebouwd te worden. In onderstaand voorbeeld wordt bij het laden een pop-upvenster geopend.

```

window.addEventListener('load', function() {
    alert('Welkom op de site!');
});

```

◆ *onunload*

Dit event treedt op indien een object niet langer is geladen. Indien dit toegepast wordt op het body-element dan geeft dit de waarde `true` terug als de gebruiker de pagina verlaat.

◆ *onresize*

Dit event treedt op als de afmetingen van het browservenster of van een frame worden aangepast.

❖ Event handlers voor alle elementen

◆ *onclick*

Dit event wordt meestal toegepast bij knopachtige elementen, afbeeldingen en hyperlinks, maar kan eigenlijk bij alle elementen gebruikt worden. De bijhorende programmacode wordt uitgevoerd wanneer de gebruiker op het element klikt. Als de handler (de JavaScript functie die hoort bij dit attribuut) de waarde `false` teruggeeft, dan voert de browser de standaardactie die bij dit element hoort *niet* uit. Er wordt bijvoorbeeld niet gesprongen naar een URL (voor het `<a>`-element), of het formulier wordt niet verstuurd wanneer op de submitknop geklikt wordt.

Ondanks het feit dat `onclick` bij alle elementen gebruikt kan worden moet hier spaarzaam mee omgesprongen worden aangezien gebruikers niet de gewoonte hebben overal op te gaan klikken.

◆ *onmousedown, onmouseup*

Deze events treden op als de gebruiker met de muis op een element staat en de muisknop respectievelijk indrukt en loslaat. De meeste elementen die `onclick` ondersteunen, ondersteunen ook deze events.

◆ *onmouseout, onmouseover*

Deze events treden op als de gebruiker met de muis over een element beweegt (`onmouseover`), of zich niet langer boven het element bevindt (`onmouseout`). Ze worden zeer veel met afbeeldingen gebruikt om een rollover afbeelding te maken, maar kunnen ook zeer nuttig zijn bij imagemaps en hyperlinks. Indien `onmouseover` bij een `<a>`-element gebruikt wordt dan zal de URL die bij deze link hoort *niet* in de statusbalk van de browser weergegeven worden als dit event de waarde `true` teruggeeft.

◆ *onmousemove*

Dit event treedt op als de gebruiker de muis beweegt als ze zich boven het element bevindt.

❖ Event handlers voor formulieren

◆ *onchange*

Dit event hoort enkel bij de elementen `<input>`, `<select>` en `<textarea>`. Het treedt op als de gebruiker de waarde van deze elementen wijzigt.

◆ *onsubmit, onreset*

Deze events horen bij het `<form>`-element en treden op als het formulier verstuurd (`submit`) of gewist (`reset`) wordt. Als de handler de waarde `false` teruggeeft, dan wordt de form *niet* verstuurd of gewist.

◆ **onfocus**

Dit event treedt op als een element wordt geselecteerd en kan toegepast worden bij hyperlinks `<a>`, bij afbeeldingsgebieden `<area>` en de formulierelementen `<input>`, `<select>` en `<textarea>`. Je kan de focus verplaatsen met de tab toets of met de muis.

◆ **onblur**

Dit event treedt op als de selectie van een element ongedaan wordt gemaakt en is dus de tegenhanger van `onfocus`.

◆ **onkeydown**

Dit event treedt op als een toets ingedrukt wordt op het toetsenbord.

◆ **onkeyup**

Dit event treedt op als de toets van het toetsenbord terug wordt losgelaten.

◆ **onkeypress**

Dit event treedt op na het `onkeydown`-event, dus als de toets ingedrukt gehouden wordt. `KeyPress` werkt enkel voor printbare karakters, dus niet voor bijvoorbeeld de alt- of ctrl-toets.

14.3 Pagina's dynamisch maken

❖ Methods en properties van window

Als je een webbrowser start, opent slechts één venster (window). Terwijl je surft op het net, kan het gebeuren dat nieuwe vensters geopend worden en andere gesloten. Dit kan je bekomen met de `window.open()` en `window.close()` - methodes van het `window` object. Deze methodes worden dus gebruikt voor nieuwe vensters, dit in tegenstelling tot `document.open()` en `document.close()` die ingrijpen op het huidige venster.

Een nieuw venster openen kan met de `open()` -methode.

```
window.open()
```

Het sluiten of beëindigen van een venster kan met methode `close()`.

```
window.close()
```

Naast het openen en sluiten kan je met JavaScript ook vensters verschuiven en van grootte veranderen. Dit kan naar een absolute waarde of met een relatieve aanpassing ten opzichte van de huidige waarde.

```
moveTo(x, y)
moveBy(dx, dy)
resizeTo(width, height)
resizeBy(dwidth, dheight)
```

Bij het openen van een nieuw browservenster kan je verschillende parameters meegeven.

```
window.open(url, name, features, replace)
```

Argument	Betekenis
url	De URL van de locatie die moet geopend worden. Als een lege string wordt meegegeven ("") wordt een leeg document getoond.
name	De naam van het venster. Deze naam kan gebruikt worden als de waarde van een <code>target</code> -attribuut bij een <code><a></code> of een <code><form></code> -tag.

features	Een string die de eigenschappen van het venster bevat. <ul style="list-style-type: none"> • height: de hoogte van het venster • left,top: de X-Y coördinaten van het venster (voor Internet Explorer) • location: de adresbalk (yes/no) • menubar: de menubalk (yes/no) • resizable: mag het venster van grootte veranderen (yes/no) • screenX, screenY: de X-Y coördinaten van het venster (voor Firefox). • scrollbars: de scrollbars (yes/no) • status: de statusbalk (yes/no) • toolbar: de toolbar met de Back en Forward knoppen enz. (yes/no) • width: de breedte van het venster
replace	Indien true, dan wordt een extra entry in de history lijst van de browser aangemaakt.

Je kan ook testen of een bestaand `window`-object reeds gesloten is, door te controleren wat de waarde is van de property `window.closed`. Dit uitgezonderd, kan je op een gesloten venster niet langer methodes of properties oproepen.

Beschouw het volgende voorbeeld. De pagina bevat een link die een nieuw venster opent en een andere link die het venster kan sluiten. Let op in de code hoe de eigenschappen van het venster bepaald worden.

```
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="UTF-8">
  <title>Dynamische pagina's</title>
</head>
<body>
<h1 id="dummy">Welkom op de site</h1>
<p>Meer informatie, klik <a href="#dummy" id="linkOpen">hier</a>.</p>
<p>Klik <a href="#dummy" id="linkSluit">hier</a> om het venster te sluiten.</p>
<script>
  document.getElementById("linkOpen").addEventListener("click", show);
  document.getElementById("linkSluit").addEventListener("click", dispose);

  var nieuwVenster;

  function show() {
    var eigenschappen = "width=300, height=300, left=150";
    nieuwVenster = window.open("", "info", eigenschappen, false);
    var nieuweBody = nieuwVenster.document.getElementsByTagName("body")[0];
    var title = document.createElement("h1");
    title.innerHTML = "Hallo!";
    var paragraph = document.createElement("p");
    paragraph.innerHTML = "Dit venster bevat allerlei info";
    nieuweBody.appendChild(title);
    nieuweBody.appendChild(paragraph);
  }

  function dispose() {
    if (nieuwVenster) {
      nieuwVenster.close();
    }
  }
</script>
</body>
</html>
```

De functie `show()` opent een nieuw venster met de naam "info" en de eigenschappen "width=300, height=300, left=150". Dit venster wordt toegekend aan de globale variabele

nieuwVenster. Er worden nieuwe html-elementen aangemaakt en aan de body van het nieuwe venster toegevoegd.

De twee links hebben een href-attribuut dat wijst naar een anker binnen de pagina. Dit anker zit op het begin van de pagina, dus eigenlijk doet dit niets. Interessanter zijn de twee event handlers op onclick.

De tweede functie is de dispose()-operatie die de close()-methode van het venster oproept.

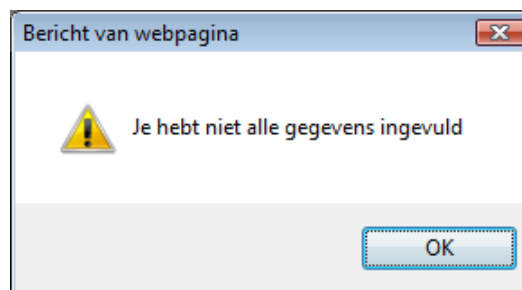
❖ pop-upvensters: alert, prompt, confirm

Je hebt in JavaScript ook de mogelijkheid om pop-upvensters te gebruiken om te communiceren met de gebruiker.

◆ *Het alert venster*

Het alert venster toont een korte boodschap die de gebruiker enkel kan bevestigen door op OK te klikken. Het wordt meestal gebruikt voor waarschuwingen. De methode heeft als argument een string die de boodschap of waarschuwing bevat.

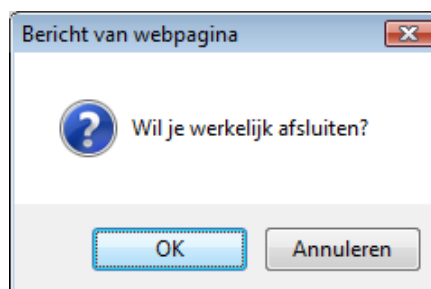
```
window.alert('je hebt niet alle gegevens ingevuld');
```



◆ *Het confirm venster*

Indien een vraag moet gesteld worden waarop bevestigend of ontkennend dient geantwoord te worden, is een confirm venster de geëigende keuze. In het pop-up venster kan de gebruiker enkel op "OK" of "Annuleren" klikken. De methode retourneert true of false, waarop je dan beslissingen kan baseren. Het argument van de functie bevat de gestelde vraag.

```
var status = window.confirm('Wil je werkelijk afsluiten?');
// status bevat true of false
if (status) {
    doe_iets();
} else {
    doe_iets_anders();
}
```

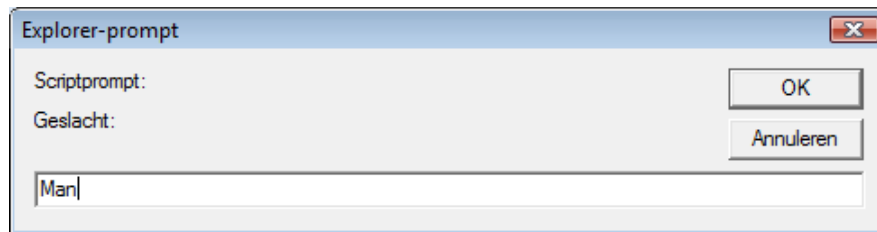


◆ *Het prompt venster*

Als je input wilt vragen doe je dat met de prompt methode van het window object. Het eerste argument van de functie is de gestelde vraag, het (optionele) tweede argument de standaardwaarde als de gebruiker op "Annuleren" zou klikken. De functie retourneert de ingetikte

waarde of de standaardwaarde waarop je dan beslissingen kan baseren. Indien die er niet zou zijn wordt null teruggegeven.

```
var antwoord = prompt(' Geslacht: ', ' Man' );
if (antwoord !== "Man") {
    document.getElementById("greeting").innerHTML = "Wel kom! ";
} else {
    document.getElementById("greeting").innerHTML = "Gegroet! ";
}
```



Toch is hier een waarschuwing op zijn plaats. Hoewel deze pop-upvensters toelaten om te communiceren met de gebruiker, moet je hiermee niet overdrijven. Gebruikers ervaren teveel pop-ups immers als erg hinderlijk, wat de gebruiksvriendelijkheid van de site niet ten goede komt. Bovendien kan de gebruiker de browser zo configureren dat pop-upvensters niet meer getoond worden, zodat je (belangrijke) boodschap toch verloren gaat.

14.4 DOM-scripting

In DOM-scripting wordt vooral gebruik gemaakt van de `getElementById()`-methode uit het W3C-DOM om HTML-elementen te benaderen. Elk HTML-element wordt als een knooppunt (node) beschouwd binnen het model.

❖ Benaderen van eigenschappen (properties)

De typische properties die kunnen benaderd worden zijn:

- `x.innerHTML`: de HTML-code binnen de HTML-container
- `x.nodeName`: het `name`-attribuut van een HTML-element
- `x.nodeValue`: het `value`-attribuut van een HTML-element
- `x.parentNode`: de parent node van een HTML-element
- `x.childNodes`: de child nodes van een HTML-element
- `x.firstChild`: het eerste kind van een knooppunt
- `x.lastChild`: het laatste kind van een knooppunt
- `x.attributes`: de attributen van een HTML-element

In deze opsomming staat `x` voor een knooppunt (bijvoorbeeld: `document.getElementById('knop')`.)

De `innerHTML`-eigenschap is een zeer handige manier om HTML-elementen aan te passen. Hiermee kan je immers de inhoud van een HTML-container opvragen en aanpassen. Het opvragen van de inhoud van een knooppunt, kan als volgt.

```
tekst=document.getElementById("paragraaf").innerHTML;
```

Voor het aanpassen gebruik je de volgende code:

```
document.getElementById("paragraaf").innerHTML="Hallo, <big>Wereld</big>";
```

De tekstuele inhoud van een knooppunt kan opgevraagd worden met de property `nodeValue`. Dit werkt enkel voor pure tekst en attribuutknooppunten. In tegenstelling tot `innerHTML` wordt hier dus geen opmaakinformatie (HTML-codes) mee opgenomen en wordt de tekstuele inhoud ook als een child beschouwd. Een voorbeeld zal veel duidelijk maken.

```
<ul id="lijst"><li>een</li><li>twee</li><li>drie</li><li>vier</li></ul>
```

In bovenstaand voorbeeld is elk `li`-element een child van de lijst. De teksten een, twee, drie en vier zijn op hun beurt ook weer een child van het list item. Als je bijvoorbeeld de waarde drie wenst terug te krijgen, dan dien je volgende code te gebruiken.

```
tekst =  
document.getElementById('lijst').childNodes[2].firstChild.nodeValue;
```

Wil je de waarde vier terugkrijgen kan dat bijvoorbeeld met:

```
tekst = document.getElementById('lijst').lastChild.firstChild.nodeValue;
```

Aangezien de witruimtes (spaties en enters) ook knooppunten kunnen zijn in de DOM-tree is het nodig in de HTML-code alles aan elkaar te schrijven om dit werkend te krijgen. Deze werkwijze wordt dan ook voornamelijk bij XML gebruikt.

❖ Het gebruik van DOM Methods

Zoals al besproken in het onderdeel over het W3C DOM zijn er verschillende methodes beschikbaar om een HTML-element te benaderen.

- `x.getElementById(id)`: selecteert op het `id`-attribuut
- `x.getElementsByTagName(name)`: selecteert op het `name`-attribuut
- `x.getElementsByTagName(tag)`: selecteert op elementnaam
- `x.getElementsByTagName(class)`: selecteert op classnaam
- `x.querySelector(selector)`: selecteert één css-selector
- `x.querySelectorAll(selector)`: selecteert alle css-selectors
- `x.appendChild()`: voegt een onderliggend knooppunt toe
- `x.removeChild()`: verwijdert een onderliggend knooppunt
- `x.createElement()`: voegt een HTML-element toe
- `x.createTextNode()`: voegt een tekstueel knooppunt toe

Als we bij het voorbeeld van de lijst bijvoorbeeld een vijfde item wensen toe te voegen kan dat met onderstaande code:

```
var knooppunt=document.getElementById('lijst');  
knooppunt.appendChild(document.createElement("li"));  
knooppunt.lastChild.innerHTML="vijf";
```

Met de `appendChild`-methode wordt een kind toegevoegd, maar aangezien dit kind opnieuw een HTML-element is wordt de `createElement`-methode gebruikt, waarna hierin met de `innerHTML`-property de tekst vijf geplaatst wordt.

Indien enkel de tekst vijf aan de webpagina moest toegevoegd worden, dan kan je dat doen met de `createTextNode`-methode.

15 Bronnen

- Eloquent JavaScript, second edition: A Modern introduction to programming. Marijn Haverbeke (<http://eloquentjavascript.net/index.html>)

W3Schools JavaScript

16jQuery

16.1 Inleiding

jQuery is een van de meest gebruikte Javascript libraries voor client-sided scripting in webapplicaties. De library wordt voornamelijk gebruikt om elementen in de DOM-boomstructuur te selecteren en te wijzigen, om event-handling en animaties te voorzien en om met Ajax te werken.

Een van de grote voordelen is dat binnen jQuery een eenvoudige syntax gebaseerd op CSS selectors gebruikt wordt om elementen te selecteren. Wat in JavaScript geschreven wordt als

```
document.getElementById('boven')
```

wordt in jQuery

```
jQuery('#boven') of nog korter $('#boven')
```

Een tweede groot voordeel ligt in het feit dat jQuery een groot aantal problemen in verband met browserinconsistenties achter de schermen verhelpt.

jQuery functies worden aangeroepen met `$(document).ready` na het laden van de DOM. Er is dus een verschil met `window.addEventListener("load" ...)`, want in dat geval wordt de functie pas uitgevoerd als de volledige pagina geladen is, dus inclusief figuren, enzovoort.

```
$(document).ready(function() {
    ...
});
```

In eerste "Hello World!" voorbeeld (jQuery_vb01.html) illustreert het gebruik van jQuery. In een eerste `script`-container wordt de jQuery library (jquery.js) als een extern bestand geladen. In de volgende `script`-container wordt de gewenste code aangeroepen van zodra de pagina geladen is in de browser. De code zorgt ervoor dat de inhoud van het element met `id 'output'` geselecteerd en aangepast wordt via de methode `html`. De uitgebreidere uitleg over de verschillende methodes om met de DOM in interactie te treden volgt later.

```
<!DOCTYPE html >
<html >
  <head>
    <title>jQuery voorbeeld 1</title>
    <meta charset="UTF-8">
    <script src="script/jquery.js"></script>
    <script>
      $(document).ready(function() {
        $('#output').html("<h1>Hello world</h1>");
      });
    </script>
  </head>
  <body>
    <div id="output"> </div>
  </body>
</html >
```

Hierboven wordt een anonieme functie gebruikt, maar het kan ook met een benoemde functie.

```
$(document).ready(toonHelloWorld);
function toonHelloWorld() {
  $('#output').html("<h1>Hello world</h1>");
}
```

16.2 Selectors

In vergelijking met JavaScript is het met jQuery veel gemakkelijker en gebruiksvriendelijker om een selectie te maken van één of meerdere elementen. De syntax voor een selectie is grotendeels hetzelfde als de syntax voor CSS-selectors.

In de methode jQuery wordt een String geplaatst tussen aanhalingstekens met daarin een selector als argument. Deze jQuery methode creëert een object van de klasse jQuery, dat een verzameling van resultaten bevat, dus alle elementen die voldoen aan de selector.

De methode kan gebruikt worden als

```
var resultaat = jQuery("selector");
```

Door de meeste webontwikkelaars wordt het aanroepen van jQuery afgekort met het symbool \$

```
var resultaat = $("selector");
```

In onderstaand voorbeeld worden bijvoorbeeld alle paragrafen geselecteerd die binnen een div met een id gelijk aan "main" staan.

```
var resultaat = $("div#main p");
```

Raadpleeg de jQuery API documentation op <http://api.jquery.com> voor een uitgebreid overzicht van CSS selectors, evenals <http://code.tutsplus.com/tutorials/the-30-css-selectors-you-must-memorize--net-16048>

Een zeer handige tool om jQuery selectors te testen en onder de knie te krijgen is <http://codylindley.com/jqueryselectors/>.

❖ De basis selectors

De meest eenvoudige selectors laten toe om een element, een id of een klasse te selecteren, eventueel gekoppeld aan de hiërarchie.

Selectorkarakter	Syntax	Betekenis
*	\$("*")	alle elementen
geen	\$("element")	element
#	\$("#idnaam")	id
.	\$(".classnaam")	class

Enkele voorbeelden

Selector	Voorbeeld	Betekenis
element	\$("a")	alle elementen 'a'
id	\$("#links")	het element met id 'links'
class	\$(".vet")	alle elementen met class 'vet'
combinatie	\$("div.links")	de div's met class 'links'
	\$("div#boven")	de div met id 'boven'
child	\$("div > a")	alle elementen 'a' die het kind zijn van een 'div'
descendant	\$("div a")	alle elementen 'a' die in een 'div' staan
next adjacent	\$("header + div")	alle eerste div's volgend op een header

next sibling	<code>\$("header ~ div")</code>	alle div's die volgen op een header en dezelfde parent hebben als de header.
--------------	---------------------------------	--

❖ De selectors voor attributen

Door gebruik te maken van de vierkante haken kan je heel ver gaan in de selectie van attributen.

Selectorkarakter	Syntax	Betekenis
[]	[attr]	element bevat het attribuut
[!]	[attr!]	element bevat het attribuut niet
[^=]	[attr^="..."]	attribuut start met
[\$=]	[attr\$="..."]	attribuut eindigt met
[*=]	[attr*="..."]	attribuut bevat
[~=]	[attr~="..."]	attribuut bevat het woord omgeven door spaties
[=]	[attr = "..."]	attribuut bevat de tekst of begint met die tekst gevolgd door een koppelteken

Hieronder worden enkele courante attribuutselectors geïllustreerd:

Selector	Voorbeeld	Betekenis
attribuut	<code>\$("[src]")</code>	alle elementen met attribuut 'src'
	<code>\$("input[name]")</code>	alle input's met attribuut 'name'
	<code>\$("img[src='ok.png']")</code>	alle img's met attribuut 'src' met waarde 'ok.png'
	<code>\$("img[src*='ok']")</code>	alle img's met attribuut 'src', 'src' heeft een waarde met daarin ergens de string 'ok'
	<code>\$("img[src^='ok']")</code>	alle img's met attribuut 'src', 'src' heeft een waarde die begint met de string 'ok'
	<code>\$("img[src\$='.png']")</code>	alle img's met attribuut 'src', 'src' heeft een waarde die eindigt op '.png'

❖ Selectors voor formulieren

Naast de bovenstaande selectors zijn er nog vele jQuery selectors beschikbaar die worden voorafgegaan door een dubbele punt (colon). Een aantal van deze :selectors zijn bedoeld voor het selecteren van invoervelden in een formulier.

Selector	Selectie
:button	alle invoervelden van het type button
:checkbox	alle invoervelden van het type checkbox
:checked	alle aangevinkte keuzevakken
:disabled	alle elementen die onbeschikbaar zijn
:enabled	alle elementen die beschikbaar zijn
:file	alle invoervelden van het type file
:focus	het invoerveld in focus
:hidden	alle invoervelden van het type hidden of met <i>visibility: none</i>
:input	alle invoervelden van het type input, textarea, select of button

:password	alle invoervelden van het type password
:radio	alle invoervelden van het type radio
:reset	alle invoervelden van het type reset
:selected	alle geselecteerde opties in een keuzelijst
:submit	alle invoervelden van het type submit
:text	alle invoervelden van het type text

❖ Selectors op child of type

Selector	Selectie
:first-child	eerste kind
:first-of-type	eerste van een type
:last-child	laatste kind
:last-of-type	laatste van een type
:nth-child()	het n-de kind
:nth-last-child()	het n-de laatste kind
:nth-last-of-type()	het n-de van een type
:only-child	het enige kind
:only-of-type	het enige van een type

❖ Andere selectors

Selector	Selectie
:contains()	element bevat specifieke tekst
:empty	alle lege elementen (zonder kindelementen)
:eq()	het element met de juiste index
:even	alle even elementen (0, 2, 4, ...)
:gt()	alle elementen met een index groter dan
:has()	element bevat minstens één element met specifieke tekst
:header	alle header-elementen (h1 tot en met h6)
:lang()	alle elementen met dat language attribuut
:last	het laatst geselecteerde element
:lt()	alle elementen met een index kleiner dan
:not()	alle elementen die niet voldoen aan de voorwaarde
:odd	alle oneven elementen (1, 3, 5, ...)
:parent	alle elementen die minstens één kindelement hebben
:root	het rootelement, dus altijd het html-element
:target	het doelelement
:visible	alle zichtbare elementen

16.3 Verschil met JavaScript

❖ Selectie van één element

Bij de selectie van één enkel element is de werking van `document.getElementById("id")` nagenoeg identiek aan die van `$("#id")`.

In onderstaand voorbeeld (`js_id.html`) wordt met JavaScript de eerste div gekozen op basis van zijn id en met de `innerHTML` methode wordt een paragraaf met "Hello World!" erin gezet.

```
<!DOCTYPE html>
<html>
<head>
  <title>Hello World</title>
  <meta charset="utf-8" />
  <script>
    window.addEventListener("load", function() {
      document.getElementById("een").innerHTML = "<p>Hello World</p>";
    });
  </script>
</head>
<body>
  <div id="een"></div>
  <div id="twee"></div>
  <div id="drie"></div>
</body>
```

De code van de jQuery variant (`jquery_id.html`) is zeer gelijkend in werking.

```
<head>
  <script type="text/javascript" src="script/jquery.js"></script>
  <script type="text/javascript">
    $(document).ready(function() {
      $("#een").html("<p>Hello World!</p>");
    });
  </script>
</head>
```

❖ Selectie van een reeks elementen

Het is pas bij een selectie van een reeks elementen dat het gemak van jQuery duidelijk wordt. Als we het bovenstaande voorbeeld aan willen passen om in alle div's de tekst "Hello World!" te zetten, dan is er behoorlijk meer JavaScriptcode nodig (`js_div.html`). Er is nu een for-lus nodig om doorheen de selectiereeks te gaan aangezien `getElementsByTagName()` meerdere elementen selecteert.

```
<script>
  window.addEventListener("load", function() {
    var x = document.getElementsByTagName("div");
    for (var i=0; x.length; i++) {
      x[i].innerHTML="<p>Hello World!</p>";
    }
  });
</script>
```

De jQuerycode (`jquery_div.html`) wijzigt echter amper. Indien een selector een reeks elementen selecteert, dan worden die ook allemaal aangesproken.

```
$(document).ready(function() {
  $("div").html("<p>Hello World!</p>");
});
```

16.4 Interactie met de DOM-boomstructuur

Er zijn talrijke methodes om te interageren met de DOM-boomstructuur. Hier worden er slechts enkele veelgebruikte uitgelegd, voor de volledige mogelijkheden raadpleeg de jQuery API documentatie (<http://api.jquery.com/category/manipulation/>).

❖ De methode `.html`

In eerder gegeven voorbeelden is de methode `html` geïllustreerd. Met deze methode kan de inhoud van een selectie opgevraagd en gewijzigd worden. Als je `html` aanroept zonder argument, dan geeft dit een String terug met daarin de inhoud van het eerste element in de selectie. Wanneer de methode `html` een String als argument neemt dan wordt dit argument gebruikt als inhoud van elk element in de selectie.

In `jQuery_html.html` wordt de inhoud van de `div` met id 'een' gewijzigd. De gewenste tekst wordt in de variabele `tekst` geplaatst. Vervolgens wordt tekst in de `div` met id 'een' geplaatst met de methode `html`.

```
$(document).ready(function() {
    var tekst = "Hello World!";
    $("div#een").html(tekst);
});
```

❖ De methode `.val`

Aan de hand van de methode `val` wordt de value van inputelementen uitgelezen of gewijzigd. In `jQuery_val.html` wordt deze methode gebruikt om de inhoud van een tekst invoerveld naar een `textarea` te verplaatsen.

```
$(document).ready(function() {
    var tekst = $("input:text").val();
    $("textarea").val(tekst);
});
```

❖ De methode `.css`

De methode `css` laat toe de css style van een selectie uit te lezen of te wijzigen. In `jQuery_css.html` wordt deze methode gebruikt om alle `div`'s een grijze achtergrond te geven.

```
$(document).ready(function() {
    $("div").css("background-color", "#aaa");
});
```

❖ De methode `.attr` en `.prop`

De methode `attr` laat toe attributen uit te lezen of te wijzigen. In `jQuery_attr.html` wordt deze methode gebruikt om elke `div` een `title` mee te geven.

```
$(document).ready(function() {
    $("div").attr("title", "Dit is een div");
});
```

Met deze methode kan je ook gemakkelijk controleren of een attribuut bestaat of niet. Om de inhoud van een attribuut op te vragen is er de methode `.prop`.

Indien je meerdere attributen tegelijk wil wijzigen kan dat door de attr-methode meervoudig te gebruiken.

```
$(document).ready(function(){
    $("div").attr("title", "Dit is een div");
    $("div").attr("style", "border: 2px solid black; background-color: #aaa");
});
```

Het kan ook iets geavanceerder door de verschillende attributen in JSON-object te steken. Hierin wordt telkens een naam en waarde gecombineerd, gescheiden door komma's.

```
$(document).ready(function(){
    $("div").attr(
        {
            title: "Dit is een div",
            style: "border: 2px solid black; background-color: #aaa"
        }
    );
});
```

❖ De methode .append en appendTo

Via de methode append kan inhoud toegevoegd worden aan een selectie. In jQuery_append.html wordt aan elke div eerst een paragraaf toegevoegd met de tekst "Hello". Daarna wordt binnen elke paragraaf een span toegevoegd met de tekst "Wereld!".

```
$(document).ready(function(){
    $("div").append($("<p></p>").text("Hello"));
    $("div p").append($("<span></span>").text("Wereld!"));
});
```

In jQuery_appendto.html wordt de werking van appendTo getoond. Bij appendTo enkel wordt vertrokken van een element en wordt dit element toegevoegd aan een geselecteerd element.

```
$(document).ready(function(){
    var tekst = $("<p>");
    tekst.append("Hallo").appendTo($("div"));
    var tekst = $("<span>");
    tekst.append("Wereld!").appendTo($("div p"));
});
```

Via \$("<p>") wordt een lege paragraaf aangemaakt. Daarna wordt in de paragraaf de tekst "Hallo" geplaatst en de paragraaf toegevoegd aan de div via de methode appendTo. Op identieke wijze wordt een span toegevoegd aan de net toegevoegde paragraaf.

❖ De methode .prepend en .prependTo

Min of meer het tegengestelde werking van append en appendTo, waarbij er nu vooraan toegevoegd wordt in plaats van achteraan.

```
$(document).ready(function(){
    $("div").append($("<p></p>").text("Wereld"));
    $("div p").prepend($("<span>Hallo </span>"));
});
```

❖ De methode `.addClass`, `.hasClass`, `.removeClass` en `.toggleClass`

Als je elementen wil aanpassen op basis van de classe kan je gebruik maken van `.addClass`, `.hasClass`, `.removeClass` en `.toggleClass`.

Met `addClass` kan je één of meerdere class-namen toevoegen aan je selectie.

```
$(document).ready(function() {
    $("div").addClass("groen links");
});
```

Met `hasClass` kan je een element met een bepaalde klassenaam selecteren.

```
$(document).ready(function() {
    if($("div").hasClass("groen")){
        ...
    }
});
```

Met `removeClass` kan je een bepaalde klassenaam verwijderen. Zonder waarde `removeClass()` worden alle klassen verwijderd.

```
$(document).ready(function() {
    $("div").removeClass("groen");
});
```

Met `toggleClass` wordt gecontroleerd of een bepaalde klassenaam bestaat. Als hij bestaat wordt hij verwijderd en als hij niet bestaat wordt hij toegevoegd.

```
$(document).ready(function() {
    $("div").toggleClass("groen");
});
```

❖ Verschillende methodes in een ketting

Het bovenstaande voorbeeld van de `attr`-methode waarin zowel attributen als CSS-stijl wordt aangepast kan ook herschreven worden om gebruik te maken van “chaining” (`jQuery_attr_chain.html`).

```
$(document).ready(function() {
    $("div").attr("title", "Dit is een div")
        .css("border", "2px solid black")
        .css("border-radius", "5px")
        .css("background-color", "#aaa");
});
```

Eens een jQuery-object via een selector aangemaakt is kan het gevonden resultaat verder gefilterd worden en kan de DOM boomstructuur nog doorlopen worden. Hiervoor kunnen onder andere de methoden `first`, `last`, `next` en `nextAll` gebruikt worden.

In `jQuery_filter.html` wordt de eerste `li` in het document bruin en krijgen de volgende andere kleuren via `method chaining`.

```
$('li').first().css("color", "brown")
    .next().css("color", "blue")
    .nextAll().css("color", "green");
```

De eerste `li` wordt bruin. Zijn eerste sibling wordt blauw en alle volgende siblings groen. De list items van de volgende lijsten worden niet gekleurd, want zij zijn geen `next` of `nextAll` van de `first` aangezien die sibling relaties zijn.

16.5 Event handling

De events zijn zeer identiek aan die van JavaScript. Meer info vind je in de jQuery API documentatie <http://api.jquery.com/category/events/>.

Muis	Toetsenbord	Formulier	Document/Window
.click()	.focusout()	.blur()	.load()
.dblclick()	.keydown()	.change()	.ready()
.focusout()	.keypress()	.focus()	.resize()
.hover()	.keyup()	.focusin()	.unload()
.mousedown()		.select()	
.mouseenter()		.submit()	
.mouseleave()			
.mousemove()			
.mouseout()			
.mouseover()			
.mouseup()			
.toggle()			

De algemene syntax is

```
$('elem').event(function() { ... });
```

In `jquery_click.html` wordt het click-event geïllustreerd. Wanneer de pagina geladen is (`ready`) wordt er bij het element met id 'knop' een eventhandler voorzien. Wanneer op deze knop geklikt wordt, wordt de geselecteerde waarde uit de `select` gehaald en uitgevoerd naar het `div`-element met id 'output'. Tevens wordt de knop op disabled gezet via de methode `attr`.

```
<html>
  <head>
    <script src="script/jquery.js"></script>
    <script>
      $(document).ready(function() {
        $('#knop').click(function() {
          var tekst = "Je koos voor " + $("#dept").val();
          $("#div#output").html(tekst);
          $('#knop').attr('disabled', 'disabled');
        });
      });
    </script>
  </head>
  <body>
    <div>
      <select id="dept">
        <option value="it">PXL-IT</option>
        <option value="tech">PXL-Tech</option>
        <option value="edu">PXL-Educational</option>
      </select>
    </div>
    <input type="button" id="knop" value="Bevestig" />
    <div id="output"> </div>
  </body>
</html>
```

In `jQuery_mousemove.html` wordt een voorbeeld gegeven van de eventhandling voor een `mousemove` event. De code werd overgenomen van <http://api.jquery.com/mousemove/>. Let in de code op het gebruik van de variabele `event` als argument van de anonieme functie:

```
$( "div" ).mousemove(function( event ) {
    var pageCoords = "( " + event.pageX + ", " + event.pageY + " )";
    $( "span:last" ).text("( event.pageX, event.pageY ): " + pageCoords );
});
```

Zie ook <http://api.jquery.com/event.pageX/> en <http://api.jquery.com/event.pageY/>

Ook via de methode `bind` kan een of meerdere event-handlers toegevoegd worden aan een selectie.

In `jQuery_bind.html` bevat `$(this)` de referentie naar het element dat de `mouse-enter` of `mouseleave` getriggerd heeft. Voor de werking van `toggleClass` wordt verwezen naar <http://api.jquery.com/toggleClass/>

```
$( "p" ).bind( "click", function(event) {
    var str = "( " + event.pageX + ", " + event.pageY + " )";
    $( "span" ).text("Click happened! " + str );
});
$( "p" ).bind( "dblclick", function() {
    $( "span" ).text("Double-click happened!");
});
$( "p" ).bind( "mouseenter mouseleave", function( event ) {
    $( this ).toggleClass("over");
});
```

Met `event.preventDefault()` kan je de standaard-actie van een element omzeilen. In `jQuery_prevent.html` wordt ervoor gezorgd dat de standaard-actie van een anchor (doorverwijzen naar een andere pagina) niet uitgevoerd wordt. Wel wordt er telkens een boodschap in een `div` geschreven en wordt de `div` toegevoegd aan het element met id 'log'.

```
$(document).ready(function(){
    $( "a" ).click(function(event) {
        event.preventDefault();
        $( "<div>" ).append("default " + event.type + "prevented")
                    .appendTo("#log");
    });
});
```

De methode `preventDefault` wordt dikwijls gebruikt om client-sided validatie te voorzien. Dit wordt in `jQuery_prevent2.html` geïllustreerd.

```
$(document).ready(function(){
    $( "input[type='submit']" ).click(function(event) {
        $( "input[type='text']" ).each(function() {
            if( $( this ).empty() && $( this ).val().trim() === "" ){
                event.preventDefault();
                $( this ).toggleClass("leeg")
                        .val("geef een waarde in");
            }
        });
    });
});
```

Via de methode `each` wordt elke input met attribuut `type` met waarde `text` doorlopen. Voor elke gevonden input wordt de anonieme functie aangeroepen. In deze functie wordt gecontroleerd of de input niet leeg is of enkel gevuld met spaties. Mocht de input leeg zijn dan wordt het input veld voorzien van de klasse "leeg". De input krijgt een rode kleur en gele achtergrond via die klasse.

16.6 Animaties

De methode `fadeIn` zorgt ervoor dat een element zichtbaar wordt, `fadeOut` zorgt dat het langzaam verdwijnt. Als argument van de methode kan de tijd meegegeven worden in milliseconden waarop het element getoond moet worden. In `jQuery_fade.html` wordt het element met id 'div1' in 3000 ms onzichtbaar.

```
$("#div1").fadeOut(3000);
```

Let op, het element wordt in het voorbeeld eerst via css onzichtbaar gemaakt via de methode `hide` (<http://api.jquery.com/hide/>). Ook verspringt de display property onmiddellijk van none naar block en zal de opacity van het element tijdens de 2000 ms van 0 (volledig transparant) naar 1 (niet transparant) gaan.

De methoden kunnen ook aangeroepen worden met drie argumenten. Het eerste argument is nog altijd het aantal ms, het tweede argument is de easing, en de derde defunctie die uitgevoerd moet worden als de tijd om te plaatsen verstreken is. In dit voorbeeld wordt div1 eerst in 3000 ms onzichtbaar gemaakt, daarna wordt de div met id 'div2' zichtbaar in 2000 ms.

```
$(document).ready(function() {
    $('#div1').click(function() {
        $(this).fadeOut(3000, "swing", function() {
            $("#div2").fadeIn(3000);
        });
    });
});
```

Via de methode `animate` kan een aangepaste animatie voorzien worden. Deze animatie gebeurt door de css-eigenschappen aan te passen. In `jQuery_animate.html` wordt de div met id 'div1' in 1000 ms van volledig ondoorzichtig (*opacity* 1) naar volledig transparant geplaatst (*opacity* 0) via `toggle`, van links verschoven over 20 px (`+=20`) en van volledige hoogte (ingesteld in de css) naar hoogte 0 geplaatst via `toggle`.

```
$("#div1").animate({
    opacity: "toggle",
    left: "+=20",
    height: "toggle"
}, 1000);
```

Het eerste argument van de methode `animate` is een object. Tussen de accolades wordt een object aangemaakt met eigenschappen voor *opacity*, *left* en *height*.

De methode `animate` kan ook aangeroepen met drie argumenten, het derde argument is de functie die uitgevoerd moet worden als de animatie klaar (`complete`) is.