1 Inhoudsopgave

2 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 **CCS3**, 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>`, `<li>`, `<body>` of `<h2>`.

3 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 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(huisttijl.css); /* invoegen van de huisstijl */
/* hierna kommen alle stijlen voor de supportpagina */
```

5 De waterval en overerving

In de term “cascading style sheets” verwijst de term “cascade” naar het watervalsysteem 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.

6 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 {
    eigenschap1: waarde;
    eigenschap2: waarde;
    eigenschap3: 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 klasseregel 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.

```
a:link      {color: #0000FF; text-decoration: none;}
a:visited   {color: #0000AA; text-decoration: underline;}
a:hover     {color: #0000FF; text-decoration: underline;}
a:active    {color: #0000AA; text-decoration: underline;}
```

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

❖ 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.

7 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
```

8 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-weigth: 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;
```

```
}
```

en 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 */
```

9 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.

10 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 medianaam 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 */
```

11 Opmaak

12 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, cyaan = 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.

13 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">inhoud</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 */
}
```

14 Het Box-model

Elk blokelement kan verder opgemaakt worden door talrijke stijldeclaraties die invloed hebben op de alineëring, 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 rechterrand 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 alineëring, de witruimte
rondom, de marges, de aan- en afwezigheid van een kader en de vorm en kleur
hiervan.</p>
<h1>het Boxmodel</h1> …
```

CSS	waarde	voorbeeld
margin-top	bovenmarge	margin-top: 200pt
margin-right	rechtermarge	margin-right: 25%
margin-bottom	benedenmarge	margin-bottom: 0.5cm
margin-left	linkermarge	margin-left: 35mm
Margin	verkorte schijfwijze	margin: 2cm margin: 2cm 3cm margin: 2cm 4cm 3cm 1cm
padding-top	witruimte boven	padding-top: 6em
padding-right	witruimte rechts	padding-right: 6em
padding-bottom	witruimte beneden	padding-bottom: 6em
padding-left	witruimte links	padding-left: 6em
Padding	verkorte schijfwijze	padding: 15px
border-top-width	kaderrand boven thin | medium | thick of in eenheden	border-top-width: thick

border-right-width	kaderrand rechts thin | medium | thick of in eenheden	border-right-width: thin
border-bottom-width	kaderrand beneden thin | medium | thick of in eenheden	border-bottom-width: 5px
border-left-width	kaderrand links thin | medium | thick of in eenheden	border-left-width: medium
border-width	verkorte schijfwijze	border-width: 5mm 3mm 2mm 6mm
border-color	aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white en yellow + rgb colors	border-color: blue border-color: #ff55cc
border-style	dotted | dashed | solid | double | groove | ridge | inset | outset	border-style: double
border-top	bovenboord: width color style	border-top: 5mm border-top: 5mm blue double
border-right	rechterboord: width color style	border-right: 3cm border-right: 3cm blue double
border-bottom	onderboord: width color style	border-bottom: 15em border-bottom: 15em blue double
border-left	linkerboord: width color style	border-left: 6px border-left: 6px blue double
border	verkort: width color style	border: 6px blue double
outline-color	zelfde kleuren als border-color	outline-color: black
outline-style	zelfde stijlen als border-style	outline-style: solid
outline-width	thin | medium | thick of in eenheden	outline-width: 4px
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.

15 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 alineëring.

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 “verwijderd” en vervolgens een nieuwe border-top toegevoegd (en eventueel border-bottom voor shaduw-effect).

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

16 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;
}
```

17 Lijsten

18 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 `<ol>…</ol>` (*ordered list*) element. Voor ongenummerde is er de `<ul>…</ul>` (*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 `<li>` (*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 `<ol>`-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.

❖ Ongenummerde lijsten

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

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

```
<body>
<h3>Dit is een ongenummerde 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 `<li>` is `value`. Het `type`-attribuut van `<ul>` en `<li>` 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>
```

19 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>
```

20 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 opsommingtekens 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;
}
```

21 Afbeeldingen

22 Bestandsformaten voor het web

Er kunnen verschillende afbeeldingformaten 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 weggegooide 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.

23 Het invoegen van afbeeldingen

❖ Het <img>-element

Het plaatsen van een afbeelding kan door middel van de open tag `<img />`. De sluitslash is niet langer nodig in HTML5, maar hier wordt gekozen deze toch aan te houden, maar `<img>` 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.

```
<img src="beelden/beeld1.gif" alt="hier staat een beeld" />
```

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 `<img />`, 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, evanals 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>
    <img src="beelden/beeld1.gif" alt="beeld 1" />
    <img src="beelden/beeld2.gif" alt="beeld 2" />
    <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 `<img>`-element. Dit is nooit doorgebroken door de ondermaatse browserondersteuning. Binnen HTML5 is van deze vervanging geen sprake meer. Je blijft dus best `<img />` 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 `data`-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 `heigth` 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>
  <img src="figuren/school.jpg" alt="Hogeschool PXL" />
  <object data="figuren/school.jpg" type="image/jpeg"></object>
  <figcaption>Foto ingevoegd met img-element (links) en met
    object-element (rechts)</figcaption>
</figure>
```

24 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>
  <img src="figuren/gebouw.jpg" alt="gebouw" id="links"  />
  <img src="figuren/gebouw.jpg" alt="gebouw" id="rechts" />
  <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.

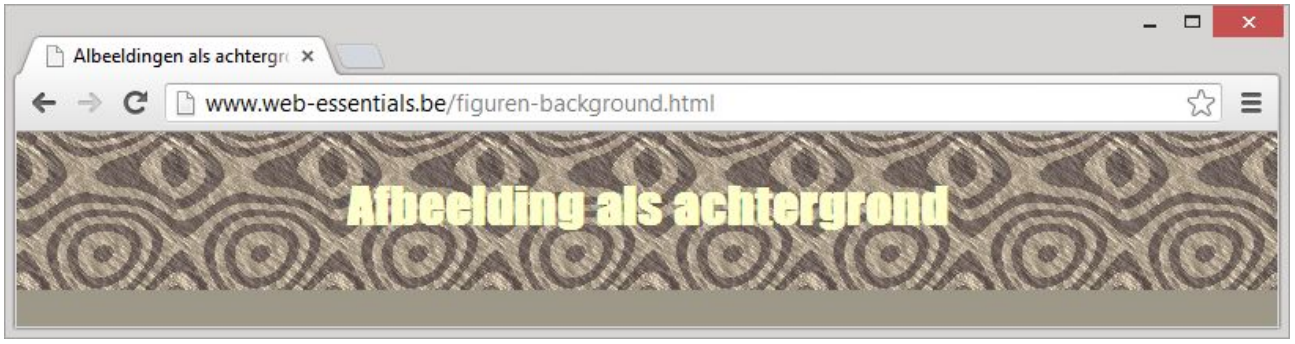
```
h1 {clear: both;}
```

❖ 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* specifieert 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 achtergrondafbeelding. 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 `<img />` opneemt binnen de koppeling-tag `<a>…</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="…"><img src="…" alt="…" /></a>
```

❖ Image maps

Om aan te geven dat het bij een afbeelding om een image map gaat, moet je aan de `<img>`-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.

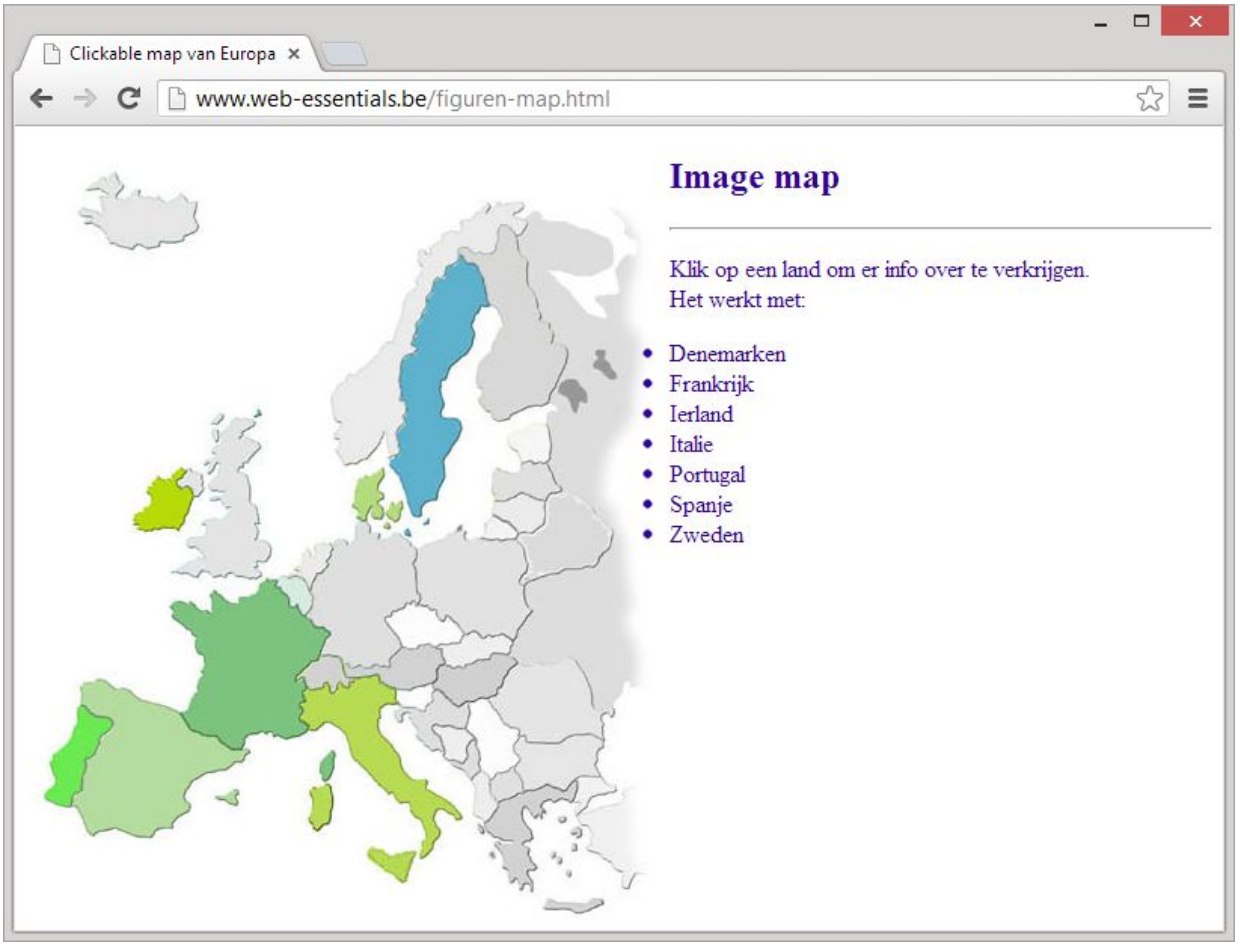
Een `map`-element bevat één of meer `area`-elementen. Elke area legt van een aanklikbaar gebied in de afbeelding de vorm, de coördinaten en de bestemming vast.

```
<img src="beelden/map.jpg" alt="map Europa" width="285" height="190"
usemap="#map" />
<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.



25 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 uitrekken 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 saven 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 saven. 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

26 Audio en video

27 Geluid

❖ Geluidsbestanden

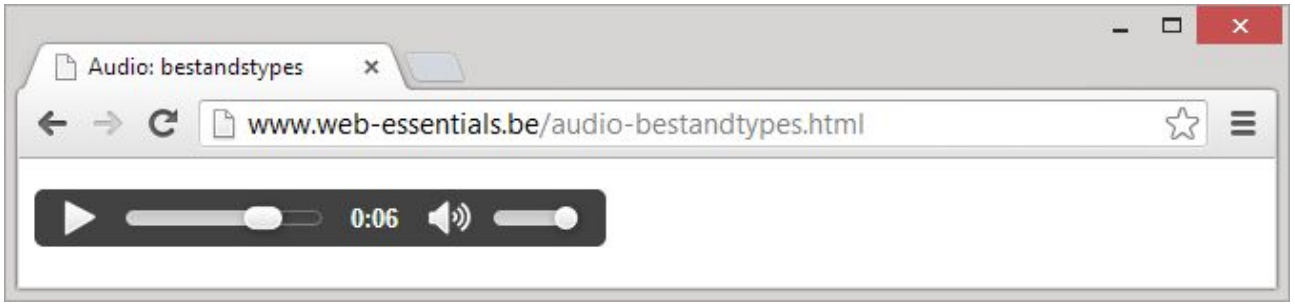
Binnen HTML5 worden er nog drie bestandsformaten voor geluid ondersteund, Ogg Vorbis, MP3 en WAV.

- **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
autoplay	speelt het geluidsbestand bij het laden van de pagina.
controls	toont de controleknoppen.
loop	speelt het geluidsbestand in een lus.
preload	laadt het geluidsbestand als de webpagina geladen wordt.
src	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>
```

28 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 attiburen 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.ogg" type="video/ogg" />
  Je browser ondersteunt geen video.
</video>
```

29 Andere interactieve inhoud

Naast geluid en film kunnen nog meer multimediabestanden 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 multimediabestanden (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
data	De locatie waar het object zich bevindt.
form	Het formulier waartoe het object behoort.
height	De hoogte van een afbeelding (in pixels of %).
name	Naam.
type	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”, …
usemap	Voor client-side clickable maps.
width	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 gespecifieerd is bij `name`. Property waardes hebben geen betekenis in HTML; hun betekenis wordt bepaald door het object zelf.

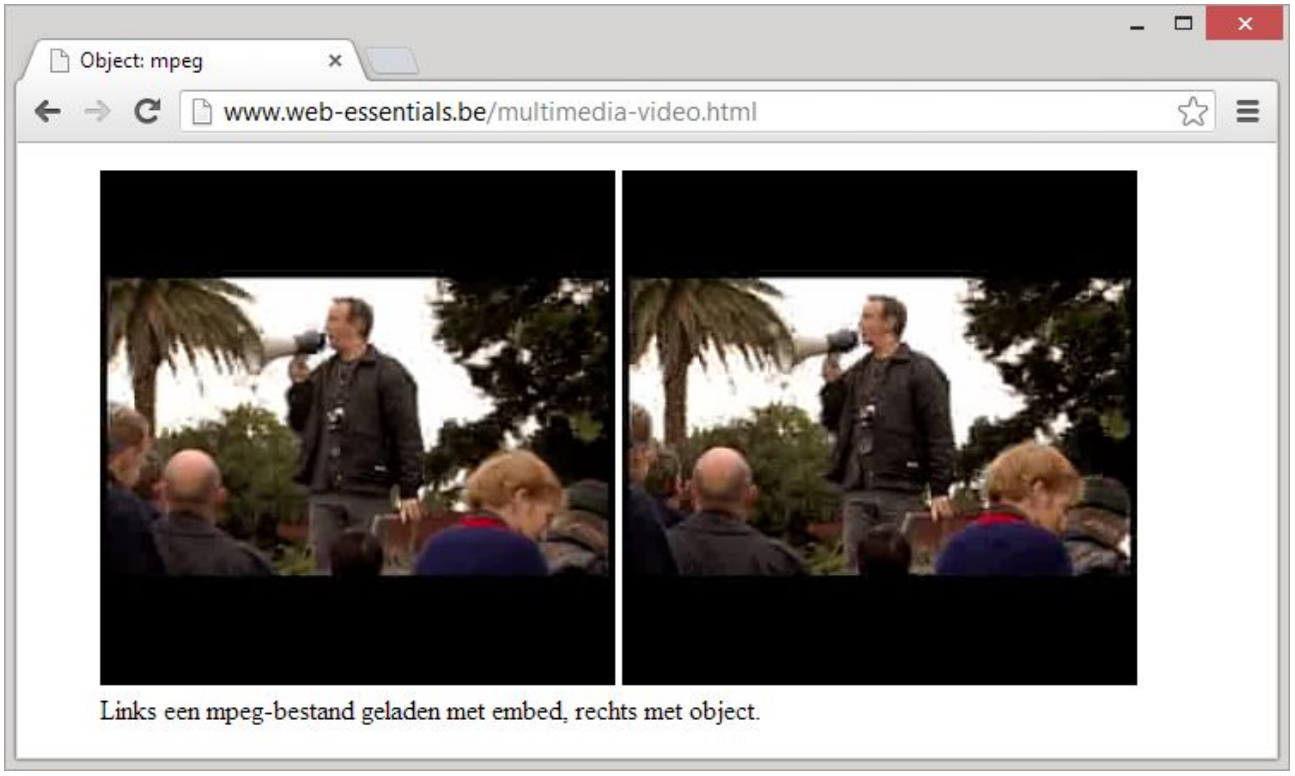
```
<object data="animatie.swf">
  <param name="height" value="40" />
  <param name="width" value="40" />
</object>
```

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"`.

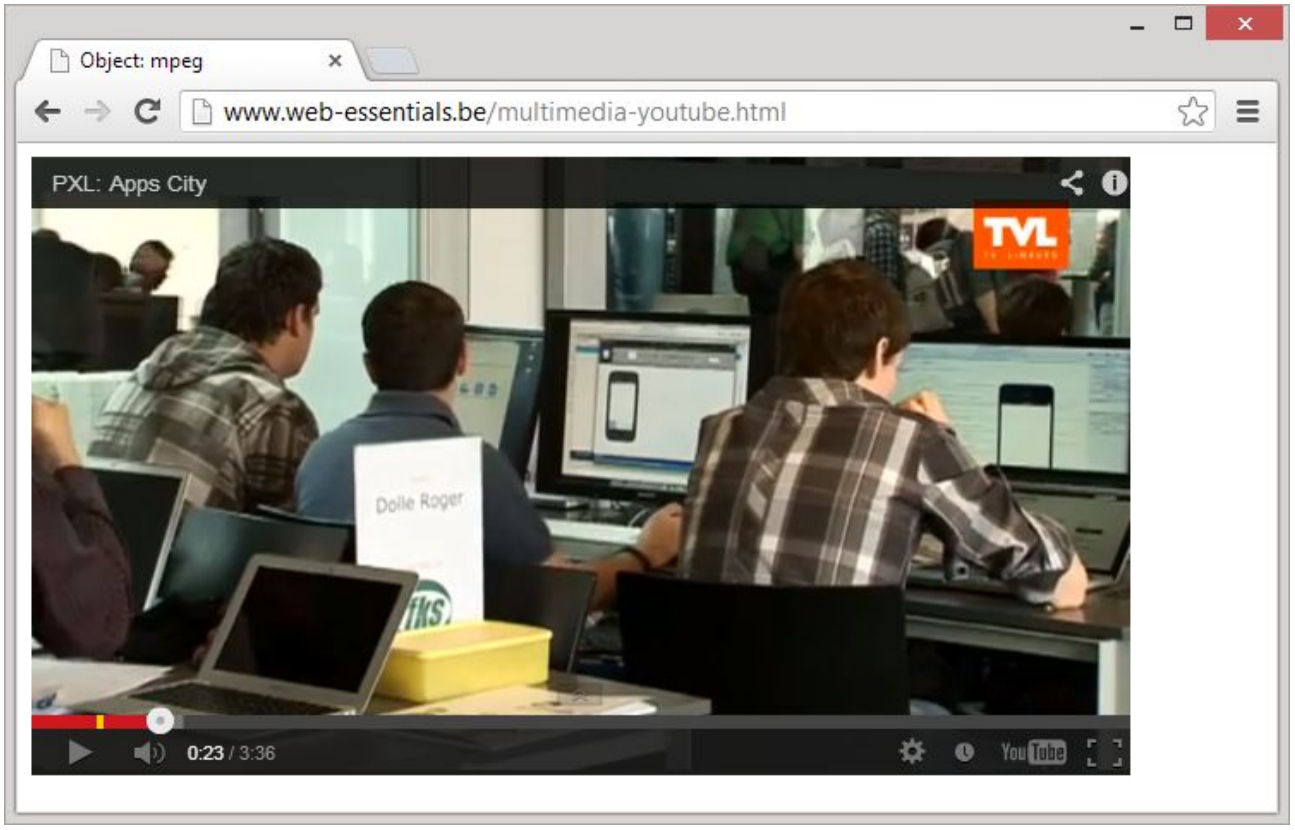
```
<object data="filmpje.mpg" type="video/mpg" width="…" height="…">
   <param name="scr" value="fimpje.mpg" />
</object>
```



❖ 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" />
```


Object: mpeg
www.web-essentials.be/multimedia-youtube.html
PXL: Apps City
Dolle Roger
0:23 / 3:36
YouTube