Webdesign

cursusjaar 2009-2010

Practicumoefeningen met

(X)HTML en CSS

Departement Informatica
Bacheloropeliding Informatiekunde
Hans Voorbij (2010)

Inhoud

Voor de oefeningen met CSS heb je een PC nodig met een werkende internet-verbinding.

1. Basisoefeningen met HTML

Basistechnieken
Dit practicum reikt je basisinformatie aan over HTML (= HyperText Markup Language) en CSS (= Cascading Stylesheets). De fijne kneepjes moet je zelf uitvinden.

We zullen in dit practicum oefenen met basistechnieken om een webpagina te coderen.
Als je nog nooit zelf webpagina's met HTML hebt gemaakt, raden we je dringend aan de basisoefeningen in deze paragraaf "met de hand te schrijven", d.w.z zonder een speciale HTML-editor. Gebruik gewoon Notepad (Kladblok), een ASCII-texteditor die onder START> PROGRAMS> ACCESSORIES op elke Windows machine te vinden is. Alleen zo leer je echt werken met de codes die achter webpagina's "schuilgaan". Als je met de belangrijkste codes vertrouwd bent geraakt, kun je tot besluit van deze paragraaf met het programma Dreamweaver enkele oefeningen herhalen (Dreamweaver is beschikbaar op alle PC's in de practicumzalen, als onderdeel van Adobe CS4). Je zult dan beter begrijpen wat voor soort hulp een goede HTML- en CSS-editor je biedt en hoe je ermee werken kunt. Als je hulp nodig hebt bij Notepad of Dreamweaver, kun je een beroep doen op de student-assistenten. Vooral doen!

Wie al wel zelf HTML-pagina's geschreven heeft, raden we aan met een HTML-editor te werken en onderstaande oefeningen daarmee te maken. Waar in de oefeningen naar Notepad verwezen wordt, kun je desgewenst "Dreamweaver" lezen.

HTML (HyperText Markup Language) is de taal die gebruikt wordt om webdocumenten te maken. Het is een standaard documenttaal, uitgevonden door Tim Berners-Lee en verder ontwikkeld door het World Wide Web Consortium (W3C), dat veel webgerelateerde standaarden uitbrengt.

- Hypertext is electronische tekst voorzien van hyperlinks naar meer tekst-informatie (of andersoortige content) binnen datzelfde of een ander document.
- Markup staat voor symbolen of codes of tags die aan een bestand worden toegevoegd, zodat een browser "weet" wat er weergeven moet worden en hoe.

Een browser is een computerprogramma waarmee je van link naar link (en terug) kunt gaan in een hypertext. Op het World Wide Web gebruiken we webbrowsers. De bekendste voorbeelden hiervan zijn momenteel Mozilla's Firefox en Microsoft's Internet Explorer (= IE). We laten het helemaal aan je smaak en voorkeur over welk van beide browsers je gebruikt – beide zijn beschikbaar op de computers in de practicumzalen – als je maar werkt met Firefox versie 2.0 of hoger, dan wel IE 6.0 of hoger. In de tekst hierna spreken we in principe alleen nog van "browser".

Bedenk wel dat alle oefeningen en opdrachten die je maakt, moeten werken in de browser. Maar wat in Firefox werkt, werkt niet per sé ook in IE, en omgekeerd...! Dit komt doordat de HTML-standaard niet op dezelfde wijze in beide browsers geïmplementeerd is. Het is een voortreffelijke gewoonte je uitwerkingen in beide browsers te testen en zo aan te passen dat ze in beide browsers werken.

Documentstructuur
Een HTML document bestaat uit twee belangrijke delen: de *head* en de *body*. Ze worden omgeven door de tags <html> </html>.
De head bevat informatie over het document (meta-informatie), zoals de titel, de gebruikte html-versie, de gebruikte codering voor lettertekens en symbolen, e.d.
De body bevat de inhoud, het gedeelte dat door de browser weergegeven wordt.
In zijn meest complete vorm wordt het HTML-document voorafgegaan door een DOCTYPE-aanduiding: een verwijzing naar een (extern) document waarin de syntax vastgelegd is die voor jouw document toegestaan is. De browser gebruikt deze informatie bij het weergeven van je document. Wij werken bij Webdesign met de DTD-versie "xhtml 1.0 strict" . Die is geschikt voor schone code, zonder speciale markup. Deze versie is ook heel geschikt om te gebruiken in combinatie met Cascading Stylesheets, die we later behandelen.

De documentstructuur wordt door elementen aangegeven (zie volgende sectie) en ziet er als volgt uit:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Documenttitel</title>                    (de document-titel komt in de titelbalk)
  </head>
  <body>
                                    (Documentinhoud:  wat getoond wordt in de webpagina)
  </body>
</html>
```

In alle voorbeelden hieronder zullen we het gedeelte

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

niet meer voluit herhalen. We volstaan met de weergave

```
<html …>
  <head>
```

In je eigen html-documenten moet je de volledige code wel aanbrengen!

Elementen

Elementen beginnen altijd met <…..> en eindigen (bijna) altijd met </……>. Op de stippeltjes tussen de spitse haakjes staat informatie in de vorm van symbolen, codes of tekst. Die spitse haakjes plus de informatie noemt men tags, zij vormen samen de markup van een HTML-document. Met behulp van tags worden de elementen in een document aangeduid, gedeclareerd.

Aan sommige begintags kunnen ook attributen met een waarde toegevoegd worden, om extra informatie over dat element mee te geven. Bijv.: <body bgcolor="#00FF00">: de achtergrond van de **BODY** is een kleur, de waarde #00FF00 levert op: groen).

Nesting

Elementen kunnen binnen andere elementen voorkomen, dit verschijnsel heet nesting. Je zet de begin- en eindtags van het binnenste element beide binnen het omvattende element, zoals in dit voorbeeld:
<buitenste><binnenste>inhoud</binnenste></buitenste>.
In een aantal gevallen is dat bij HTML zelfs verplicht, bijv.
<html><head></head><body></body></html>

Leer jezelf aan nesting altijd consequent toe te passen, om onaangename verrassingen (lees: incorrect werkende HTML-code) te voorkomen.

Opdracht 1 Een simpele HTML-pagina maken

Open de tekst-editor Notepad, en type een tekst in naar het hiervolgende voorbeeld. De **vette tekst** is wat je moet intypen, de *cursieve* is uitleg over wat de code doet. Bij ***cursief en vet*** moet je de informatie intypen die er gesuggereerd wordt.

<html …>	*(html-tekst begint, info over html-versie e.d.)*
<head>	
<title>Persoonsgegevens</title>	*(pagina-titel die in de browserbalk komt)*
</head>	
<body bgcolor="#FFFF80">	*(begin hoofd inhoud,attribuut bgcolor (achtergrondkleur) met als waarde #FFFF80, een kleur geel)*
** Persoonsgegevens **	*(: paragraaf begint & eindigt)*
** **	*(: break, sla een regel over)*
** Naam: *Jouw naam* **	
** Woonplaats: *Jouw Woonplaats* **	
** Studentnummer: *Jouw Studentnummer* **	
</body>	*(eind hoofdinhoud)*
</html>	*(einde html-tekst)*

- Sla dit bestand op als studentnummer-test.html. Let op dat je het bestand opslaat als ‘text’’. Open het vervolgens in je browser om te zien of het goed gegaan is. Je ziet, alles wat er uiteindelijk wordt afgebeeld, is hetgeen er in de <body> tussen de verschillende tags <….> … </…> staat.

- Een paar nieuwe tags: Maak de layout van de tekst zo dat “Persoonsgegevens” weergegeven wordt met tag <h3> (header3) in plaats van , en zorg dat de woorden Naam, Woonplaats en Studentnummer behalve als ook nog vet afgebeeld worden, en de andere 3 woorden cursief.

<h3>	</h3>	*(geeft tekst weer in “lettertype” Header 3)*
		(em= “emphasized text”; hiermee wordt in onze browsers tekst cursief afgebeeld; <i> kan ook, maar werkt niet in alle browsers goed)
		(zo wordt tekst vet afgebeeld; <b> kan ook, maar werkt niet in alle browsers goed)

Sla je bestand weer op en bekijk het in je browser.

Dit was om een indruk te krijgen waar we het eigenlijk over hebben; een simpel “plat” tekstbestand met informatie over hoe iets afgebeeld moet worden, die vervolgens ‘gelezen’ wordt door een browser. HTML pagina’s kunnen natuurlijk ook met een HTML editor gemaakt worden, die je “ helpt”, en ook nog volledig WYSIWYG (What You See Is What You Get) gemaakt worden, net zoals je tekst opmaakt in bijv. Word. We zullen hier evenwel voor het begrip van de materie met de hand blijven coderen.

Opdracht 2 Een lijst maken

Soms is het eleganter om gegevens in een lijstje te presenteren. HTML biedt hiervoor enkele eenvoudige mogelijkheden. Een lijst bestaat uit een of meer items. We proberen eerst de ongeordende lijst. De volgorde van de items in zo'n lijst is in principe willekeurig; de items worden voorafgegaan door een bullet.

<html …> *(html-tekst begint, info over html-versie e.d.)*
<head>
<title>Lijst Persoonsgegevens</title> *(pagina-titel die in de browserbalk komt)*
</head>

<body bgcolor="#FFFF80"> *(begin hoofd inhoud,attribuut bgcolor (achtergrondkleur) met als waarde #FFFF80, een kleur geel)*
** Persoonsgegevens ** *(: paragraaf begint & eindigt)*
<ul> *(<ul> : unordered list begint)*
**
Naam: *Jouw naam*
** *(<li>: list item begint & eindigt)*
**
Woonplaats: *Jouw Woonplaats*
**
**
Studentnummer: *Jouw Studentnummer*
**
**
** *(: unordered list eindigt)*
</body> *(eind hoofdinhoud)*
</html> *(einde html-tekst)*

- Sla dit bestand op als studentnummer-lijst1.html. Bekijk het resultaat in je browser.

Als de items in een lijst een strikte volgorde moeten hebben, maken we het beste gebruik van een geordende lijst, een ordered list, aangeduid met de tags <ol> .... De items in de lijst worden automatisch genummerd. Probeer dit uit:

- Verander in studentnummer-lijst1.html de <ul> en tags door <ol> ... /ol>. Sla het bestand op als studentnummer-lijst2.html. Bekijk dit bestand in je browser en vergelijk het resultaat met studentnummer-lijst1.html.

Opdracht 3 Een tabel maken

Het resultaat van opdracht 1 kan nog overzichtelijker dan in een lijst worden gepresenteerd. We zullen de zojuist gemaakte data in een tabel plaatsen. Maak een tabel van 3 rijen en 2 kolommen. De (basis) HTML-tags voor tabellen zijn als volgt.

<table> *(tabel openen)*
<tr> *(rij openen – rij duidt horizontale cellen in een tabel aan)*
<td> *(kolom openen – kolom slaat op de verticale cellen in een tabel)*
</td>
</tr>
</table>

Merk op dat in HTML de inrichting van een tabel in principe van links naar rechts verloopt. Je maakt dus eerst een rij en daarna de kolommen binnen die rij.

Sla je net gemaakte bestand op, maar nu onder de naam studentnummer-pers.html. Zet de 3 regels met jouw data (dus niet het woord Persoonsgegevens) in een tabel met rand dikte 1. De tabel moet 50% van het browserwindow beslaan. Hieronder is het begin voorgedaan, maak het zelf af, en sla je bestand weer op en bekijk het in je browser.

<html …>

<head>
<title>Persoonsgegevens</title>
</head>

<body bgcolor="#FFFF80">

<h3>Persoonsgegevens</h3>
**

**

<table width=50% border="1"> *(open tabel, rand dikte 1, breedte tabel 50%=halve window)*
<tr> *(openen tabel-rij)*
<td> *(openen kolom)*
** Naam: ** *(inhoud 1e kolom 1e rij oftewel de 1e cel dus)*
</td> *(sluiten 1e cel)*
<td> *(openen 2e kolom)*
** *Jouw naam*: ** *(inhoud 2e cel)*
</td> *(sluiten 2e cel)*
</tr> *(sluiten 1e rij)*
<tr> *(openen 2e rij.....*
<td>
etc etc etc *(maak de rest van de tabel zelf)*
</td>
</tr> *(sluiten laatste rij)*
</table> *(sluiten tabel)*

</body>

</html>

Opdracht 4 Hyperlinks maken

Denk je aan Internet, dan denk je aan hyperlinks: connecties naar andere documenten op het internet of naar andere plaatsen in het huidige document. Links kunnen voorkomen in de vorm van tekst of een afbeelding.

We zullen nu kijken hoe we een link coderen, maar dan moet er natuurlijk wel iets zijn dat we kunnen linken. Daarom, neem de HTML-pagina van opdracht 3 en sla hem op onder de naam studentnummer-overig.html.
Verander in dit nieuwe document het woord “persoonsgegevens” in: “overig”. Verander de woorden “ naam” , “woonplaats” en “studentnummer” in: “vooropleiding”, “studie” , “hobby’s”. Vul rechts ervan deze velden in (hobby’s etc.), en sla de pagina weer op.
We hebben nu dus 2 pagina’s met een tabel, een keer met persoonsgegevens, en een keer met “overig”. Deze 2 pagina’s gaan we aan elkaar linken. We gebruiken de volgende codes:

De “anchor”-tag (<a>...) wordt gebruikt om een hyperlink te maken.
Met het attribuut “href” wordt de bestemming van de link aangegeven.
De waarde ervan zal de naam van een ander document zijn, en/of de naam van een anchor in hetzelfde document met een hekje (#) ervoor.

Zo zal de volgende code:

```
<a href="bestemming.html">Dit is een link</a>
```

een link genereren die zal leiden naar het bestand bestemming.html.
Let er op dat hier wordt verwezen naar bestemming.html in dezelfde directory (bestemming.html heeft geen “pad” informatie erbij). Dit noemen we een relatieve URL (Uniform Resource Locator).
Je kunt ook absolute URLs als waarde meegeven, bijvoorbeeld naam/adres van een server gevolgd door het pad naar het bestand en eventueel de naam van het bestand. De volgende link is absoluut: absolute link

Open de pagina studentnummer-pers.html (persoonsgegevens). We gaan een link maken naar de pagina studentnummer-overig.html (overig). Verander het stukje:

<h3>Persoonsgegevens </h3>

in:

<h3>Persoonsgegevens Overig</h3>

Sla op. Tussen de tags <h3> en </h3> stond al het woord “Persoonsgegevens”, nu ook het woord “Overig”. En we hebben een link (die wordt standaard blauw onderstreept) gemaakt van het woord “Overig”. Bekijk het in je browser, het moet er alsvolgt uitzien:

Door op “Overig” te klikken word je naar de “overig” pagina gebracht (studentnummer-overig.html). Nu, als je op “overig” bent, is er nog geen manier om terug te keren naar “persoonsgegevens”, want er is nog geen link gemaakt in de “overig” pagina. Die ga je nu maken op dezelfde manier als boven beschreven, maar dan andersom.
Sla de pagina weer op en bekijk de 2 pagina’s in je browser. Als alles goed is heb je nu 2 pagina’s waartussen je kunt ‘springen’ d.m.v. de gemaakte links.

Opdracht 5 Andere content dan text opnemen

Natuurlijk zijn er ook andere zaken te presenteren in een HTML-pagina dan alleen tekst. Foto’s, geluidsbestanden, animaties, of video om er maar een paar te noemen.

Zoek online een jpg-afbeelding (bestandnaam*.jpg) of een afbeelding in gif- of png-formaat. Sla die op in dezelfde directory als je HTML-bestanden.

Open weer je eerste pagina, studentnummer-test.html, en voeg in onder de tekst:

<img src="*afbeelding.jpg*" alt="" border="0">

Breng in bovenstaande tag de naam aan van je afbeelding (uiteraard inclusief de juiste extensie: ….jpg, gif of png) en kijk of het werkt. Geef bij alt=” “ een beknopte beschrijving van de afbeelding.

Van een afbeelding kun je ook een link maken.

2. Oefeningen met CSS

HTML en CSS

In HTML kunnen we onderscheid maken tussen fysieke elementen en attributen, en logische elementen en attributen. Fysieke elementen en attributen kunnen de manier waarop een bepaald element wordt weergegeven wijzigen. Te denken valt aan lettertype, kleur en positie. De overige elementen noemen we logisch, ze hebben te maken met de structuur van informatie (tabellen, paragrafen, body, headers etc.) of met links.

Merk op dat bijvoorbeeld heading elementen (zoals we zagen bijv.: <h3>) geen fysieke elementen zijn, hoewel ze toch van invloed zijn op de weergave van dat tussenkopje. Dit komt omdat browsers zelf standaardregels hebben die de weergave van koppen en kopjes betreffen.

Vaak wil men meer invloed hebben op het uiterlijk van webdocumenten, maar de fysieke elementen in HTML bieden slechts zeer beperkte mogelijkheden om een webdocument op te maken. Het statement *Building Web pages with HTML is like painting a portrait with a paint roller* is misschien wat vergezocht, maar geeft aan waar het probleem zit. Historisch gezien is dit goed te verklaren. HTML is ooit opgezet als een taal om een logische structuur te bieden aan wetenschappelijke artikelen en deze uit te wisselen via Internet .
Maar de eerste versie van HTML beantwoordde onvoldoende aan de behoefte om informatie te visualiseren. Daarom werden in latere versies font tags met attributen als grootte en kleur in het leven geroepen.

Desondanks biedt HTML te weinig mogelijkheden voor adequate representatie van informatie. Alleen door inhoud (logische elementen, “content”) en opmaak (fysieke elementen, ”weergave-stijlen”) van elkaar te scheiden kan een adequate representatie worden bereikt.

Met behulp van CSS (Cascading Style Sheets) kunnen we inhoud en opmaak van een document scheiden: inhoud in HTML, en opmaak in CSS. CSS is een techniek die veel controle biedt over de opmaak van webdocumenten en ook het onderhoud en de consistentie van opmaak vergemakkelijkt, zeker als je een website met veel pagina’s hebt.

CSS biedt een schat aan mogelijkheden om typografie, lay-out en positionering te declareren. Ook biedt het andere voordelen: werken met CSS impliceert bijvoorbeeld kleinere, overzichtelijke HTML documenten en makkelijker te onderhouden consistente vormgeving van een site.

We zullen in dit stuk de basisprincipes van uiterlijk tot positionering behandelen m.b.v. stukken uit online tutorials. Het is erg wijs (zeker als je nog nooit hebt gewerkt met CSS) om deze tutorials te maken. Lezen van materiaal is een eerste, ermee omgaan een tweede!

Zie deze syllabus als een leidraad en zoek voor details op de aangegeven websites. In plaats van deze syllabus te volgen is het natuurlijk ook mogelijk boeken te kopen over CSS.

Een goed boek is *Cascading Style Sheets: The Definitive Guide, Third Edition*, door Eric Meyer (O’Reilly). Ook handig als naslagwerk voor de rest van je studie!

We zullen veel gebruik maken van tutorials van www.w3schools.com waarin voor vele talen tutorials worden aangeboden. Wil je een echte expert worden in CSS, is het aan te raden de gehele tutorial door te werken.

Nog een opmerking vooraf: gebruik van een programma als Dreamweaver CS4 is ten zeerste aan te raden omdat d.m.v. kleuren in HTML en CSS-bestanden goed wordt aangegeven wat de verschillende elementen zijn en hoe hun verhouding tot elkaar is.

Het begin

Een van de grootste krachten van Cascading Style Sheets is (de naam zegt het al) het concept “cascading”. De term is ontleend aan cascade, een waterval waarbij het water trapsgewijs van rots op rots valt en aldus een aaneengesloten geheel vormt. Bij cascading krijgt het hogere (meer specifieke) niveau prioriteit boven het lagere (algemenere) niveau.
Stel dat je een bepaalde stylesheet toepast op een heleboel web pagina’s. Nu kan het zijn dat je van duizend pagina’s binnen slechts één enkele pagina alle koppen groen wilt maken. Dan kun je simpelweg ingeven in die ene pagina (ook al link je een andere stylesheet eraan) dat je alle headings groen wilt hebben; zoiets als: h1,h2,h3 {color:green}. Cascading zorgt er voor dat de gelinkte (algemene) regels overschreven worden.

Dit is niet alleen handig voor web-designers. Ook lezers van webpagina’s kunnen veel profijt hebben van CSS. Lezers kunnen een aangepast stylesheet, bv. voor een braille-regel, laten toepassen op alle websites die ze bezoeken, door dit in de browser mee te geven.

Als je meer wilt lezen over de geschiedenis van CSS en de (tekstuele) principes klik dan op:

http://www.w3.org/Style/LieBos2e/history/

Inline en linked

Een stylesheet kan op verschillende manieren gekoppeld worden aan een HTML pagina; de style regels kunnen worden opgenomen in een html bestand of beschreven in een aparte (css-)file die *gelinkt* wordt aan de html file.

Inline

Inline bestaan er twee typen:

1. Een stijlblok in een HTML pagina kan er als volgt uitzien (zet het onderstaande tussen de <head> </head> tags):

```
<style TYPE="TEXT/CSS">      (begin CSS)
p                            (selector p)
{                            (openen eigenschappen)
font-family : Arial;         (lettertype)
font-size : 12px;            (lettergrootte 12 px)
letter-spacing : 10px;       (ruimte tussen de letter 10 px)
}                            (sluiten eigenschappen)
</style>
```

Nu worden deze style regels op het gehele lokale html bestand uitgevoerd. Zou je deze regels willen uitvoeren op meerdere html pagina’s, dan zul je dus ook elke keer deze <style> tags met hun inhoud moeten kopiëren.
NB. Als je meerdere attributen gebruikt, worden deze van elkaar gescheiden met een punt-komma (;)

2. Je kunt ook inline binnen het html bestand iets *ter plekke* veranderen voor die ene tag met behulp van het attribuut *style*. Dan ziet het er als volgt uit:

```
<p style="letter-spacing:inherit">content content content</p>
```

Extern (linked)

Extern (in een apart bestand) linken van een stylesheet heeft grote voordelen. Je hoeft al je elementen maar éénmaal te definiëren/vorm te geven en kunt het stylesheet linken aan diverse html pagina’s. Als er dan iets veranderd moet worden (bv. al jouw headings level 1

<h1> moeten rood gekleurd worden), dan hoeft dit maar eenmaal te worden veranderd, in de externe stylesheet, en wordt dit doorgevoerd in *alle* daaraan gelinkte HTML pagina's.

Je zult je css-file (in Dreamweaver > new file > css file) moeten linken door in je html-file de volgende html code toe te voegen tussen de <head> tags:

<link href="file.css" rel="stylesheet" type="text/css" />

Je linkt de file.css aan deze html file waarbij je aangeeft dat het van het type css is. Voor "file.css" vul je natuurlijk je eigen bestandsnaam in, bijv. studentnummer_externalstyle.css.

Selectors en structuur

Cascading style sheets bestaan uit een of meer regels die omschrijven hoe een HTML element weergegeven moet worden. De basisregel ziet er als volgt uit:

Selector {eigenschap: waarde;}

(Het deel tussen accolades noemen we "declaratie")

Een selector kan op verschillende manieren worden aangesproken. De simpelste manier selecteert een element met behulp van zijn tag naam (p, h1, etc.,).

Klik op onderstaande link om meer over CSS syntax te lezen:

http://www.w3schools.com/css/css_syntax.asp

Voorbeeld

Om dit geheel goed door te hebben (selectors, declaraties en grouping), bekijk onderstaand fragment en plak het als oefening op twee manieren binnen een html file, extern en intern (zoals boven in het vorige stuk beschreven staat, oftewel met een keer proberen met <link> (extern) en met <style> (intern).

```
body {background: white; color: yellow}
h1, h2, h3, h4, h5, h6 {font-family: Helvetica, sans-serif;
                        color: white; background: black;}
h1, h2, h3 {border: 2px solid gray; font-weight: bold;}
h4, h5, h6 {border: 1px solid gray;}
p, table {color: gray; font-family: Times, serif;}
pre {margin: 1em; color: maroon;}
```

Toelichting

Een paar opmerkingen:

- h1 – h6 worden gedefinieerd. Dit houdt in dat ALLE headings binnen het html document de gedefinieerde eigenschappen krijgen.
- Zou je een eigenschap (bv. Font-family: Courier) willen toekennen aan één bepaald element binnen dat document, dan zou dat er als volgt kunnen uitzien met behulp van een html attribuut class:
  - In de CSS: p.courier {font-family: Courier}
    je kunt ook p weglaten waardoor deze klasse op ieder element kan worden toegepast. Dit zou er uit zien als .courier {font-family:Courier}
  - In de HTML: tekst
    zou je de p weglaten uit de css-code, kun je dus de class ook aan andere tags meegeven, bv. aan een body: <body class="courier">content</body>.
- Je hebt hiermee al een klein stuk CSS geschreven. Als je CSS groter wordt, is het handig om commentaar te zetten waarin je kort beschrijft wat je precies wilt met elementen. Dit

kan worden gedaan door te beginnen met /* , dan de tekst van je commentaar, en af te sluiten met een */.
- Bv. /* Dit is commentaar */

Per lespagina op http://www.w3schools.com/css/default.asp vind je welke waarden een eigenschap/attribuut kan aannemen. Bekijk deze goed en probeer ze uit!

Units en Values

Units en values kunnen worden onderverdeeld in kleuren, lengte- en percentage- waarden.

Kleur

Je kunt kleuren op verschillende manieren benoemen:

1) met namen:
```
h1 {color: maroon;}
```
2) met RGB waarden in percentage
```
h1 {color: rgb(0%,0%,0%);}  ---> dit geeft zwart
```
3) met RGB waarden in hun numerieke equivalent (0-255)
```
h1 {color: rgb(255,0,0);} ---> dit geeft rood
```

Voor verdere oriëntatie over RGB waarden, zie bv.:
http://en.wikipedia.org/wiki/RGB

Lengte

Deze waarden zijn onder te verdelen in absoluut en relatief.

Absolute lengte-eenheden zijn:
- in (inches)
- cm (centimeter)
- mm (millimeter)
- pt (punt)
- pc (pica; 1 pica is gelijk aan 12 pt)
- px (pixel)

Je kunt deze lengte-eenheden plakken achter een lengte-attribuut, bv p (font-size:12pt).

NB. Het mooie van pixel eenheden is dat ze ook negatief kunnen zijn. Dit is erg handig voor positionering, zoals verderop duidelijk wordt.

Relatieve lengte-eenheden:
Alleen de meeste gebruikte lengte-eenheid wordt hier besproken: *em*. In CSS is een *em*-waarde de waarde van de grootte van een bepaald lettertype. Dit houdt in dat als je een grootte van 12 pt meegeeft aan een lettertype, dan is 1*em* gelijk aan 12 pt. Als je aangeeft dat de tekst 1*em* van de linkerkant in je browser moet beginnen (met het attribute margin-left:1em), dan neemt de *em* de waarde 12 (1 x 12) aan.

Percentage

Percentages werken hetzelfde als in HTML, waarbij je waarden relatief schaalt aan de hand van percentages.

Bv. margin-left:10% geeft aan dat het element waar het van toepassing is 10% van de totale ruimte links moet beginnen.

CSS2 kent nog veel meer waarden. Klik op onderstaande link om meer informatie te krijgen en om voorbeelden te zien.
http://www.w3.org/TR/REC-CSS2/syndata.html#length-units

Tekst eigenschappen

Tekst kent vele eigenschappen en attributen. Volg hiervoor de voorbeelden die beschreven staan op de website w3c schools.

Over tekst:
http://www.w3schools.com/css/css_text.asp

Over eigenschappen van lettertypen:
http://www.w3schools.com/css/css_font.asp

NB. Voorbeelden die worden gegeven kun je online met de editor veranderen en je kunt on the spot het resultaat bekijken. Bedenk goed dat het hier gaat om inline stylesheets (<style> tags).

Hyperlinks

Omdat hyperlinks binnen een homepage nogal veel gebruikt worden, zullen in deze paragraaf de belangrijkste link-attributen (<a>) apart met hun waarden worden behandeld.

CSS kent 4 belangrijke klassen voor anchors (links). Deze klassen zijn een extensie op <a> en worden daarom *pseudo-klassen* genoemd.

a:link	Beschrijft een link die nog niet gebruikt is en waar je nog niet met de muis over heen bent gegaan. (Default zijn deze links in een browser vaak blauw.)
a:visited	Beschrijft een link, die aangeklikt is (en dus bezocht). (Default vaak paars in een browser).
a:hover	Beschrijft hoe een link eruit gaat zien wanneer er met de muiscursor overheen wordt gegaan.
a:active	Beschrijft hoe een link eruit gaat zien tijdens het proces van aanklikken (dit duurt dus maar heel kort).

De links kennen aardig wat attributen waarbij color en text-decoration het meest worden toegepast.

Color kent de waarden zoals hiervoor al beschreven. Text-decoration kent de volgende waarden:

Blink	Beschrijft een knipperende tekst (is irritant, dus vermijden)
Inherit	Erft de waarde van een bovenliggend element in de hiërarchische boomstructuur van een html-document. *Inherit* is een waarde die op veel plaatsen gebruikt kan worden, niet alleen bij links. Bv. Je geeft aan de <body> een achtergrond kleur mee. Als je binnen <body> een <p> element beschrijft met p{background-color:inherit), erft dat element <p> zijn waarde van het hogere element <body>.
Line-through	Geeft een lijn door de tekst
None	Geeft alleen de tekst
Overline	Geeft een lijn boven de tekst
Underline	Geeft een lijn onder de tekst

Voor een aantal voorbeelden klik op onderstaande link:
http://www.tizag.com/cssT/pclass.php

Randen en boxen

Veel designers gebruiken tabellen binnen een pagina om positionering te bewerkstelligen of om bijvoorbeeld aan een document-onderdeel (bv. een paragraaf) wat eigenschappen toe te

kennen als achtergrond, kleur of een rand. CSS biedt interessante alternatieven voor tabellen.

We beginnen nu eerst met randen. Randeigenschappen kun je in principe aan alle tags meegeven, maar je moet je wel afvragen of en wanneer dat zin heeft (bv. een paragraaf, of een hele body).

Oefeningen met randeigenschappen kun je vinden op:
http://www.w3schools.com/css/css_border.asp

Natuurlijk is het een optie (en niet per se een slechte) om tabellen[1] te gebruiken, maar het kan ook anders (met behulp van CSS). Het model waar je dan in CSS mee werkt, heet het ***CSS box model***. Dat box model gaat o.a. uit van:

- positionering van een document-onderdeel (blok) in een ruimte (canvas): de afstand van de rand van een blok tot aan de rand van het canvas wordt margin genoemd.
- de positie van de content van een blok t.o.v. de rand van dat blok: de ruimte tussen de content en de rand van het blok wordt padding genoemd.

Een simpel voorbeeld staat hier:
http://www.htmldog.com/guides/cssbeginner/margins/

Het hele boxmodel volgens het W3C, waarbij de belangrijkste attributen van margin en padding worden toegelicht, kun je vinden op:
http://www.w3.org/TR/REC-CSS2/box.html

Nu je wat attributen hebt gezien, is het tijd om wat voorbeelden te bekijken via w3schools, je kunt ze in principe allemaal bekijken en uitproberen, want positionering zul je goed moeten oefenen wil je dit in de vingers krijgen:

Voorbeelden van margins:
http://www.w3schools.com/css/css_margin.asp

Voorbeelden van paddings:
http://www.w3schools.com/css/css_padding.asp

** en -tags**

Nu je iets over positionering hebt geleerd en over opmaak, kun je jezelf afvragen hoe je bv. een compleet lege box creëert. Daarmee wordt bedoeld dat als je positionering in de vorm van padding en margin toevoegt, dit hiervoor werd toegekend aan bv. een paragraaf, maar het grote nadeel is dat een paragraaf of andere bestaande html-elementen al default eigenschappen hebben.
Wil je een box echt compleet leeg opgeleverd krijgen dan zijn daar twee andere tags voor: en . Dat deze tags in principe niets doen, kun je zien als je in een html pagina tekst toevoegt (ditzelfde geldt voor) en dit bekijkt in een browser. In principe wordt een leeg schilderdoek aangeboden, wat helemaal door een web designer kan worden ingevuld.

 heeft in principe betrekking op block level elements, in het bijzonder grote selecties, binnen een html document. heeft in principe betrekking op inline elements. Voor een verdere toelichting, zie:

http://www.mako4css.com/BasDiv.htm

Het zou dus heel handig zijn om blok elementen van een pagina (bv. een header, een navigatiebalk, een content window) met divs aan te geven. Natuurlijk is het mogelijk om

[1] Zie voor een voorbeeld van tabellen gecombineerd met CSS:
http://www.lugod.org/presentations/css/demo/tables.html

daarbinnen weer <p> toe te passen. HTML blijft een boomstructuur, waarbinnen steeds elementen als child kunnen worden geplaatst.

Belangrijke voorbeelden
Er zijn een paar belangrijke voorbeelden die je moet bekijken en uitproberen voordat je verdergaat met echte positionering. Het gaat om een aantal attributen die veel worden gebruikt voor positionering en om een aantal goede voorbeelden van CSS-gebruik.

Het attribuut float (staat hier automatisch op right; probeer eens andere waarden, te vinden m.b.v. Dreamweaver):
http://www.w3schools.com/css/tryit.asp?filename=trycss_float

Het verschil met het vorige voorbeeld is dat het plaatje nu in een zit en dus in een block element:
http://www.w3schools.com/css/tryit.asp?filename=trycss_float3

Hoe een simpele website in elkaar te zetten met een simpele header, block en footer (m.b.v. margin, padding en gecombineerd):
http://www.w3schools.com/css/tryit.asp?filename=trycss_float6

Positionering

Positionering van blokken binnen html-pagina's kan een goed alternatief zijn voor het werken met tables of frames. Positionering kan op drie manieren gebeuren:

- Relatieve positionering
  - dit betekent dat een element relatief vanaf zijn originele positie binnen de html code naar een andere wordt verplaatst. Een simpel voorbeeld vind je op: http://www.w3schools.com/css/tryit.asp?filename=trycss_position_relative
- Absolute positionering
  - met deze positionering kan een element overal op een pagina worden gezet. Een simpel voorbeeld vind je op: http://www.w3schools.com/css/tryit.asp?filename=trycss_position_absolute

Volg nu eerst (alle) tutorials op over positionering op de volgende pagina:
http://www.w3schools.com/css/css_positioning.asp

- Fixed positionering
  Deze positionering geeft een soort frames effect (wordt niet door alle browsers ondersteund).

.

Afsluiting: enkele tips
Je bent in principe klaargestoomd om je eerste website met CSS in elkaar te zetten en alle informatie te verwerken. Een paar tips ter afsluiting:

- Voordat je begint met coderen, schets eerst voor jezelf wat je nou daadwerkelijk wilt hebben als inhoud en hoe je de informatie (bv. als blokken) wilt plaatsen.
- Begin niet te ambitieus. Begin eerst met een paar (heel) simpele webpagina's en voeg dan langzaam kennis toe.
- Als je een paar professionele en volwaardige voorbeelden wilt zien, is het verstandig te kijken op www.csszengarden.com waar professionele designers verschillende CSS files hebben gemaakt aan de hand van *EEN ENKELE HTML-FILE*. Dit laat goed zien hoe uiteenlopende webproducten je met CSS kunt maken. Raak niet ontmoedigd door de voorbeelden, want het vergt veel oefening om dit te kunnen en deze voorbeelden worden ook niet van je verwacht. Maar ze kunnen je wel inspireren!
- Nuttige informatie en goed uitgewerkte voorbeelden kun je verder bijv. vinden op:
  - http://www.alistapart.com/articles/howtosizetextincss/

3. Valideren

Tot slot een heel belangrijke verwijzing die je zeker moet hebben uitgeprobeerd voordat je aan de Gemeentewebsite-practicumopdracht gaat beginnen.

Op de website van het W3C (http://www.w3.org/) vind je rechts op de pagina HTML & CSS (http://www.w3.org/standards/webdesign/htmlcss) links naar een online CSS-validator en HTML-validator (Markup): parsers waarmee je kunt laten testen of je html- en css-bestanden valide zijn volgens de standaarden van het W3C.

Probeer de werking van deze validators uit.
Check jouw XHTML- en CSS-bestanden en verwerf het “valid HTML / CSS” -icon voor jouw bestanden.
Leer jezelf aan om regelmatig met deze validators te werken wanneer je een website aan het maken bent.
Ook Dreamweaver CS4 biedt goede validators voor XHTML en CSS. Maar verifieer altijd nog even of de code die door Dreamweaver valide is bevonden, ook door de W3C-validator als correct wordt aangemerkt!