WEB FRONT DEVELOPMENT

3 NMCT

DEEL II : CSS3

Multimedia en Communicatietechnologie
Kevin De Rudder – Johan Vannieuwenhuyse

howest
De Hogeschool West-Vlaanderen
Lid van de Associatie Universiteit Gent

Lid van Associatie Universiteit Gent

Graaf Karel de Goedelaan 5 - B-8500 Kortrijk (Belgium)

☎ +32 56 24 12 44
www.howest.be

Inhoudstabel

1 Introductie

1.1 Historiek

De huidige versie van CSS is CSS2.1 (zonder spatie) wat eigenlijk al een bewerkte versie van CSS2 is die in 1997 gepubliceerd is. Het bizarre aan CSS2 is dat deze eigenlijk nooit officieel gereleased werd.

De CSS working group begon al in 1998 met het bouwen van CSS3 maar doordat de frustraties van webdevelopers omwille van de inconsistentie van CSS2 tussen browser zo hoog opliep, besloot het W3C op te houden met elke nieuwe functionaliteit in CSS. In plaats daarvan werkte men aan een versie CSS2.1. In 2005 werd alle CSS3 functionaliteit terug naar draft status gebracht.

Vandaag de dag is men terug volop aan CSS3 aan het werken, meeste browser werken mee en implementeren de features.

1.2 Module status and Recommendation Process

Wanneer een bepaald document door het W3C voor het eerst geaccepteerd wordt door het W3C dan krijgt deze de status *Working Draft*. Dit betekent dat het document met zijn CSS specificaties gepubliceerd wordt zodat de community dit kan bekijken. Met community bedoelen we, browsermakers, working groups en andere geïnteresseerden. Inderdaad, indien jullie dat willen, kunnen jullie jezelf inschrijven op een mailing list waardoor je de documenten kan bekijken. Reken dan wel dat je zo'n 100 mails per dag krijgt.

Niet alle documenten halen de volgende status. Voor een document van working draft naar *last call* kan gaan kan een hele periode verstrijken. Last call betekent dat de review periode afgesloten wordt en dat men naar een volgende status gaat.

De volgende status *is Candidate Recommendation*, wat betekent dat het W3C erkent dat de specificaties van het document nuttig kan zijn. Op dit moment zijn browser zeker dat de features zullen gereleased worden. Wanneer 2 of meer browsers de properties op dezelfde manier implementeren en er geen technical issues meer zijn, kan het document overgaan tot *Proposed Recommendation*, dit betekent dat het document volwassen is en alleen door het comitee moet goedgekeurd worden zodat het een *Recommendation* kan worden.

1.3 Browsers en CSS

Net zoals bij HTML5 zijn de ene features in CSS3 wel gesupporteerd in bepaalde browsers en de andere dan weer niet. Zolang CSS3 in development is, zal je hier rekening moeten mee houden.

Wanneer een bepaalde module in CSS3 onder review staat kunnen zaken nog veranderen of zelfs nog verwijderd worden. Browsers moeten die functionaliteit wel implementeren willen we zien welk effect ze hebben op webpagina's. Elke browser kan een bepaalde functionaliteit anders interpreteren, wat tot rare situaties kan leiden. Om dit te voorkomen gebruiken de browser specifieke prefixes.

Stel u voor dat er een property berenvel zou bestaan en dat alle browser het zouden implementeren dan kunnen we volgende code schrijven:

```
-moz-berenvel: value;    /* Firefox */
-ms-berenvel: value;     /* Internet Explorer */
-o-berenvel: value;      /* Opera */
-webkit-berenvel: value; /* WebKit */
```

In theorie schrijf je code, die wanneer de property finaal geimplementeerd wordt door de browser, je deze code niet meer hoeft aante passen. Wanneer je zou werken zonder de prefixen en een feature zou klaar zijn, dan zou dit problemen kunnen geven wanneer de browser de geupdate versie implementeert.

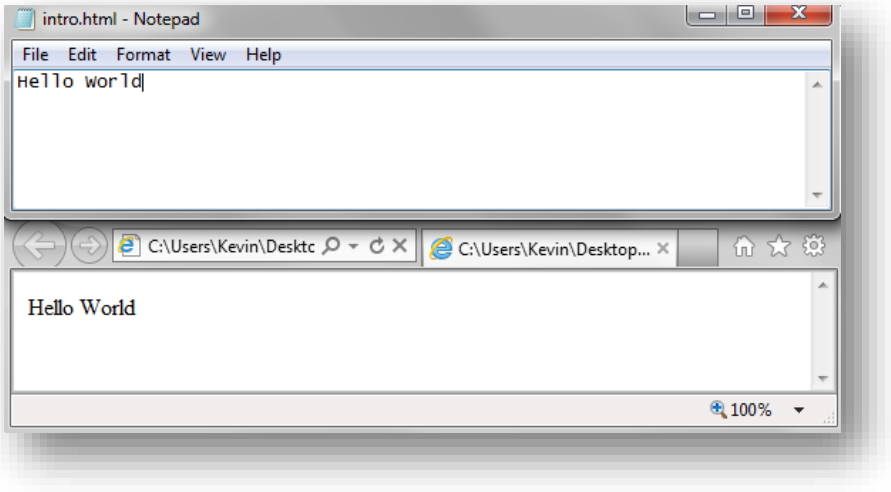
1.4 CSS layout engines

Elke browsers gebruikt zijn eigen layout rendering engine, waardoor de implementatie van een zelfde feature potentieel nog kleine verschillen kan inhouden:

- Safari en Chrome: WebKit
- Microsoft: Trident
- Opera: Presto

2 MediaQueries

Iedereen kan webdeveloper zijn. Open notepad, type “Hello World” sla de file op met extensie .htm of .html en je heb een webpagina.



Oke, dit is simpel gezegd, we moeten zorgen voor kleuren enz, maar wat we niet mogen vergeten dat we websites bouwden , die enkel bekeken werden op een desktop of laptop. Als we vandaag kijken naar de toestellen die onze bezoekers gebruiken, dan moeten we onze grenzen duidelijk verleggen.

2.1 Media types

Met CSS kan je al een tijdje verschillende media types aanspreken. We gebruiken daarvoor het media attribuut:

```
<link href="css/global.css" rel="stylesheet" media="screen" />
<link href="css/global.css" rel="stylesheet" media="print"  />
```

Er zijn 10 media types beschikbaar:

All	Alle toestellen
Aural	Speech synthesizers
Braille	Braille feedback
Embossed	Braille printers
Handheld	Mobiele toestellen
Print	Printen
Projection	Presentaties
Screen	Kleurschermen
Tty	Terminals
TV	Televisie

We hebben een breed gamma aan toestellen van schermen van 3,5 inches tot 32 inches, waardoor we met media types geen oplossing aan alle problemen kunnen bieden. Binnen eenzelfde mediatype kunnen er bijvoorbeeld diverse schermgroottes of resoluties bestaan.

In CSS3 wordt dit probleem opgelost door gebruik te maken van media queries.

2.2 Media Queries

Het grote voordeel van media queries is het feit dat ze zich baseren op de attributen en specificaties van een bepaald toestel (tablet, desktop, HTDV …), je hoeft dus geen javascript meer te schrijven om te kijken wat de grootte en resolutie van een scherm is.

2.2.1 Syntax

Een media query gaat de waarde invullen naargelang de regels die je schrijft een true opleveren. Je kan media queries op 3 manieren gebruiken:

1. Een andere file oproepen op basis van een expressie

```
<link href="global.css" rel="stylesheet" media="logic media and (expression) " />
```

2. Een externe file importeren op basis van een expressie

```
@import url('mobile.css') logic media and (expression)
```

3. De derde en waarschijnlijk meest gebruikte manier, is media queries gebruiken binnen uw stijl elementen zelf

```
@media logic media and (expression) { rules }
```

Veel van de syntax komt bekend voor. Hierbij is “media” hetgeen wat we al kennen van vroeger (media type), “expression” is nieuw en waar het allemaal om draait. Je declareert een expressie door de and operator te gebruiken in combinatie met *media features*. Dit zijn de parameters die je gebruikt binnen de expressie.

2.2.2 Media Features

Media features zijn informatie over het device waarmee je werkt:

- Afmetingen
- Resolutie
- …

Deze informatie wordt gebruikt om de expressie te evalueren. Je moet een expressie zien als:

> “gebruik deze styles wanneer het device een scherm minder dan 480 px heeft”

2.2.3 Width en Height

De meest voorkomende eigenschappen, die je gebruikt wanneer je met media queries werkt, zijn de breedte en de hoogte van het toestel. Wanneer je met een breed scherm werkt zal je verschillen secties op je scherm naast elkaar kunnen schikken, terwijl op een iphone je beter deze secties onder elkaar plaatst.

```
@media screen and (min-width: 400px)
{
        h1 {
                background: black url('landscape.jpg') no-repeat 50% 50%;
                color: white;
                height: 189px;
                margin-bottom: 0;
                padding: 20px;
            }
}
```

In dit voorbeeld wordt er enkel een image getoond wanneer je de breedte van 400px overschrijdt. Dit wil zeggen dat op kleinere schermen de image achterwege gelaten wordt omdat er te weinig plaats is.

BELANGRIJK: wanneer je met width en height werkt, gebruik altijd min-width en min-heigth omdat je nooit precies op de pixel kan werken.

2.2.4 Device width and height

Werkt hetzelfde als width en height, met dit verschil dat de device-width de grootte van het scherm geeft ipv de grootte van de browser.

Deze eigenschappen worden vooral gebruikt bij het maken van mobiele toepassingen. Volgend voorbeeld toont hoe in een container elementen naast elkaar getoond worden op een breder scherm maar onder elkaar op bijvoorbeeld een iPhone :

```
.container
{
    width: 500px;
}

.container div
{
    float: left;
    margin: 0 15px 0 0;
    width: 235px;
}

@media only screen and (max-device-width: 320px)
{
    .container
    {
        width: auto;
    }
    .container div
    {
```

```
        float: none;
        margin: 0;
        width: auto;
    }
}
```

2.2.5 Denken aan performantie

Wanneer je images in CSS files gebruikt worden die door bepaalde browsers gedownload, wat normaal is. Stel dat je nu verschillende images gebruikt voor verschillende device-width of height dan worden die allemaal gedownload. Dit zorgt er natuurlijk voor dat de pagina trager geladen wordt.

Misschien is het beter om verschillende stylesheets te maken voor grotere of kleinere toestelen:

```
<link href="desktop.css"  media="screen and (min-device-width: 480px)">
<link href="tablet.css"   media="screen and (max-device-width: 480px)">
```

Een ander voorbeeld is om een style te voorzien voor een horizontale of verticale view, wat natuurlijk nodig is bij het fysisch draaien van smart phones of tablets.

Je kan natuurlijk niet voor elke grootte of breedte een aparte style gaan maken. Hou de kerk in het midden is de boodschap.

2.2.6 Multiple Media Features

Natuurlijk kan je ook meerdere media features tegelijkertijd gebruiken. Daarvoor kan je de **and** operator gebruiken. In volgende code snippet zorgen we ervoor dat alle vormen van widescreen getarget wordt:

```
@media only screen  and (aspect-ratio: 15/10)
                    and (aspect-ratio: 16/9)
                    and (aspect-ratio: 16/10)
{
    rules
}
```

Aspect ratio wordt hier gebruikt om bepaalde breedte-hoogte verhoudingen na te gaan. Een andere mogelijkheid is gebruik maken van Pixel Ratio waar je bijvoorbeeld schermresoluties mee kan beschrijven, bijv.1024x768.

Wanneer je met multiple media features werkt kan je ook verschillende media types combineren:

```
@media all          and (orientation: landscape),
       projection   and (orientation: portrait)
{
   rules
}
```

3 Selectors

Sinds het begin van CSS spelen selectors een grote rol. CSS1 had 5 à 6 selectors, CSS2 had al 12 selectors en CSS3 bouwt verder, waarbij we tegen het einde van de rit meer dan het dubbele aan selectors zullen hebben.

3.1 CSS2 Selectors

Attribute selectors zijn al geintroduceerd in CSS2 waarbij je elementen selecteert op basis van zijn attributen zoals bijv. href of title.

Bekijken we volgende markup:

```
<ul>
    <li><a href="" lang="en-GB" rel="friend">Kevin</a></li>
    <li><a href="" lang="es-ES" rel="contact es">Johan</a></li>
    <li><a href="" lang="es-MX" rel="contact mx">Archibald</a></li>
</ul>
```

Met volgende CSS:

```
a[rel]
{
    color: red;
}
```

We selecteerden hiermee de elementen, die een rel attribuut specifieren, zonder te kijken wat er in het attribuut zit. Natuurlijk kunnen we ook tot op de attribuut waarde gaan. Dit is een voorbeeld van een *Simple Attribute Selector.*

```
a[rel=’contact es’]
{
    color: red;
}
```

Willen we beide contact attributen targetten, dan kunnen we gebruiken maken van een *Partial Attribute Selector selector.*

```
a[rel~=’contact’]
{
    color: red;
}
```

We hebben ook nog een laatste selector, nl de *Language Attribute selector*. Deze selector selecteert elementen op basis van hun *lang* attribuut:

```
a[lang|='es']
{
```

```
    color: red;
}
```

3.2 Attribute selectors in CSS3

We hebben net de attribuut selectors bekeken, die we kunnen gebruiken in CSS2. CSS3 bouwt daar natuurlijk op verder. De CSS3 selectors bieden ons nog meer flexibiliteit.

3.2.1 Substring Attribute Value Selector

De eerste selector die we bespreken kunnen we gebruiken om bepaalde waarden terug te vinden in een element, zo hebben we bijvoorbeeld de *Beginning Substring Value Selector* en de *Ending Substring Value Selector*.

In volgend voorbeeld gebruiken we de caret (^) om aan te duiden dat een bepaalde waarde van een attribuut moet beginnen met een bepaalde tekst:

```
a[href^='mailto'] { background-image: url('email_go.png'); }
a[href^='ftp'] { background-image: url('folder_go.png'); }
a[href^='https'] { background-image: url('lock_go.png'); }
```

We kunnen ook selecteren op basis van het einde van een tekst. Dit doen we via het $ teken. Een veel gebruikte situatie hier is het tonen van een bepaalde image (een icon) op basis van de extensie:

```
a[href$='.pdf'] { background-image: url('pdf.png'); }
a[href$='.doc'] { background-image: url('word.png'); }
a[href$='.rss'] { background-image: url('feed.png'); }
```

Zoals je kan raden bestaat er ook een selector die gewoon overal in de tekst kan kijken, dit kunnen we doen door te werken met een *:

```
a[title*='image'] {}
```

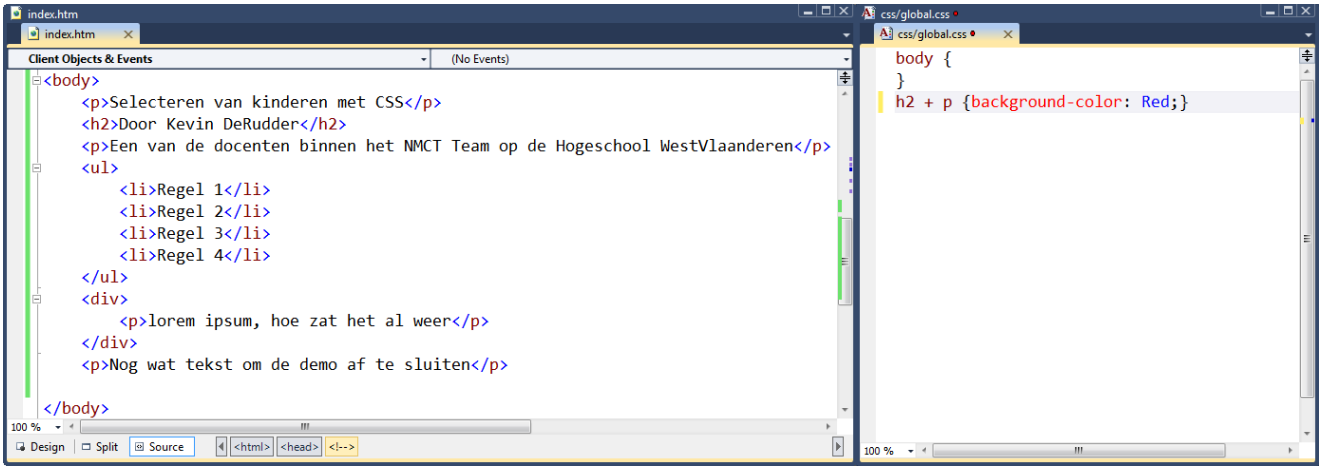
In sommige gevallen wil je de attribuut selectoren ook gaan combineren. Dit kan je door de verschillende selectoren achter elkaar te plaatsen:

```
a[href^='http://'][href*='/folder2/'][href$='.pdf'] {}
```

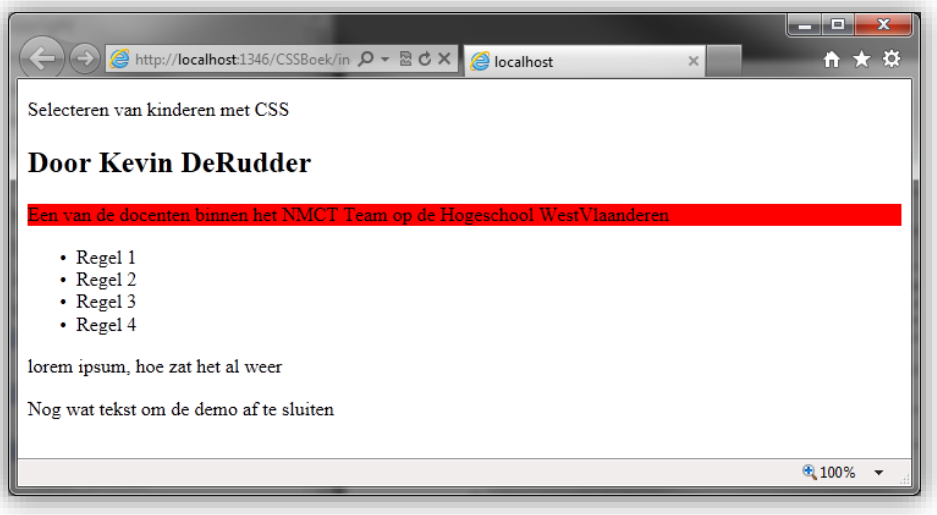
3.2.2 Selecteren van kinderen van een element

De laatste selector die we kunnen gebruiken is een *combinator*. In CSS2 heb je al een *Adjacent Sibling Combinator*. Deze selector zoekt de **eerste elementen** , die na een ander bepaald element komen en op hetzelfde niveau binnen de DOM vallen:

```
h2 + p {background-color: Red;}
```

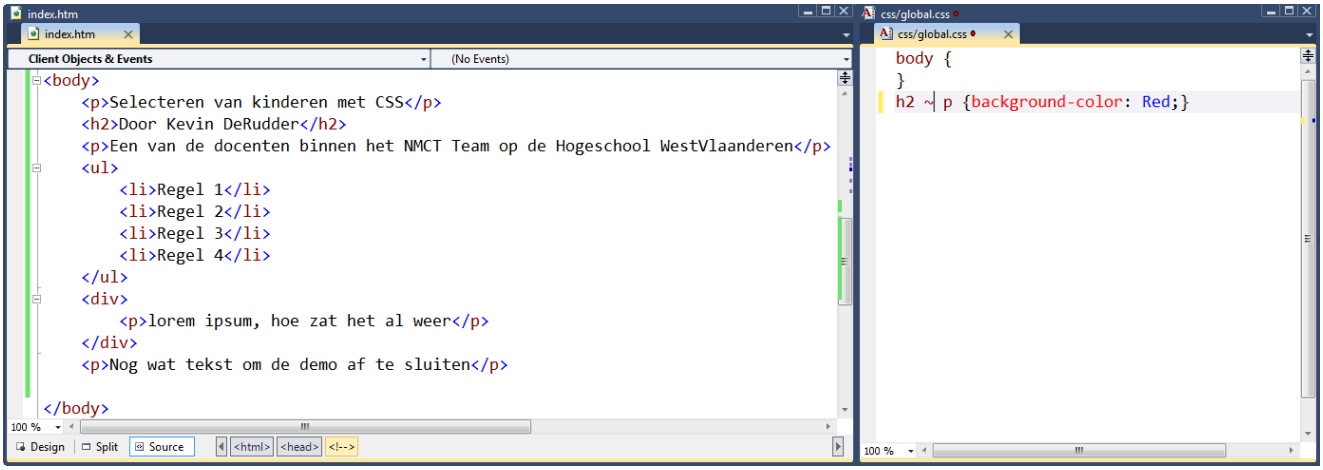


Dit geeft als resultaat de eerst p's , die op dezelfde hoogte van h2 staan en erop volgen:

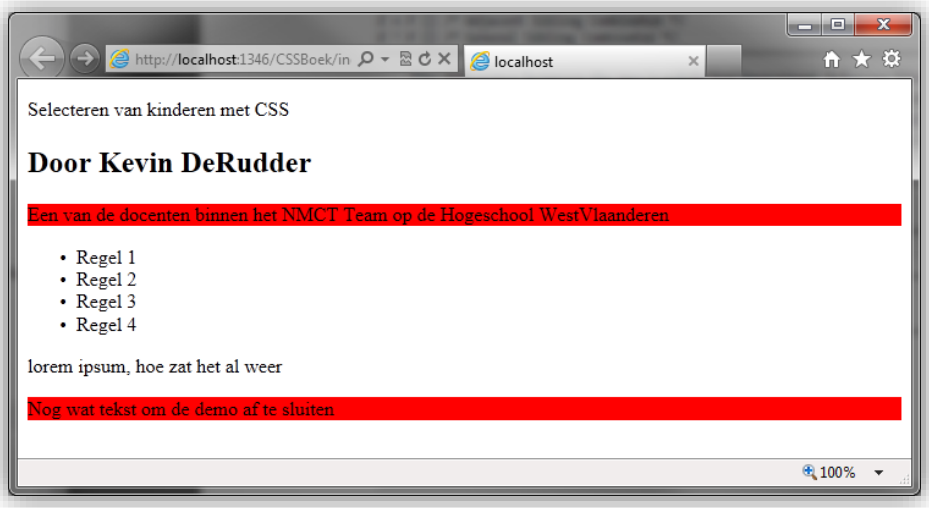


In CSS3 kunnen we gebruik maken van een combinator die **alle elementen** selecteert op hetzelfde niveau in de DOM tree.

```
h2 ~ p {background-color: Red;}
```



Met als resultaat alle p's , volgend op h2 en opdezelfde hoogte ervan :



3.3 Pseudo-classes en Pseudo-elementen

Sinds de eerste CSS versie werd er al over *pseudo-elementen* en *pseudo-classes* gesproken. Dit zijn selectors die je toelaten om bepaalde stijlen toe te passen op basis van een toestand van een element:

```
a:hover{}
a:visited{}
a:active{}
```

3.3.1 Structural pseudo-classes

Een pseudo-class laat toe om een element te selecteren op basis van informatie, die niet als waarde of als tekst terug te vinden is in de DOM tree. Structurele pseudo classes filteren op basis van de positie in de DOM structuur.

Bekijken we volgende voorbeelden:

```
element:first-child { background-color: Orange; }
```

```
element:last-child { background-color: Orange; }
```

3.3.2 nth-* pseudo-classes

We kunnen ook werken met de index van een element in de DOM tree. De syntax die we hiervoor gebruiken is de *nth-*. We selecteren hiermee een element op basis van zijn positie binnen de DOM tree startend van een bepaalde parent.

In zijn meest simpele vorm kan je bijvoorbeeld het tweede element selecteren:

```
element:nth-child(2){ background-color: Orange; }
```

Deze syntax wordt pas echt interessant als we de even elementen kunnen selecteren. Hiervoor gebruiken we de *n* na ons getal. Deze n moet niet worden gedeclareerd, maar duidt op een reeks van natuurlijke getallen. De expressie wordt als volgt gevalueerd:

Met n als waarde 0: 2*0 = 0 (element bestaat niet)
Met n als waarde 1: 2*1 = 2 (het tweede element)
Met n als waarde 2: 2*2 = 4 (het vierde element)
enz...

```
element:nth-child(2n){ background-color: Orange; }
```

Gaan we nog een stap verder kunnen we het volgende bekomen

```
element:nth-child(2n + 1){ background-color: Orange; }
```

Achter de schermen worden volgende berekeningen gemaakt:

2*0 + 1 = 1

2*1 + 1= 3

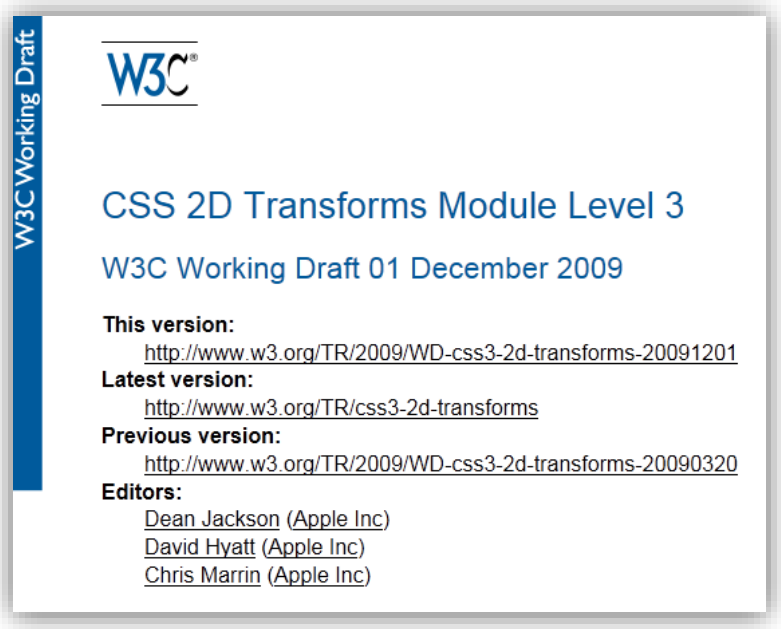
2*2 + 1 = 5 (het vierde element)

…

4 2D Transformations

Wanneer we bepaalde items in onze pagina's willen tranformeren of animeren hebben we daar meestal (meerdere) images of scripting voor nodig.

In 2008 heeft het webkit team een voorstel ingediend zodat items kunnen getransformeerd worden vanuit CSS:



Deze module is aangenomen door het W3C in 2009. Het leuke aan de feature is dat die momenteel door meeste browser geïmplementeerd is.

4.1 Transform Property

2-dimensionele transformaties kan je uitvoeren op elementen door gebruik te maken van de transform property. Via deze property zijn er een aantal functies beschikbaar. Elke functie maakt gebruikt van 1 enkele value of een comma-seperated list van values.

Daarnet werd vermeld dat alle browsers de transformaties implementern, maar je moet wel nog gebruik maken van de prefixes. Vaak wordt ook aangeraden, om een extra lijn toe te voegen zonder prefix, zodat de functionaliteit ook in toekomstige browser releases blijft werken.

```
Element {
    -moz-transform: function(value);    /* Firefox */
    -ms-transform: function(value);     /* Internet Explorer */
    -o-transform: function(value);      /* Opera */
    -webkit-transform: function(value); /* WebKit */
    transform: function(value);         /* Future-proofing */
}
```

4.2 Transform Origin

4.2.1 Rotate Transform

Roteert een object met een gegeven hoek startend vanuit het transform origin punt.

5 Oefening 1 : Nieuwe tags in CSS

5.1 Doelstellingen:

- Word-wrap
- Border-radius
- HSLA
- Linear-gradient
- Box-shadow
- Tekst-shadow
- Transform

Bronmateriaal voor deze oefening kan je vinden in **OefeningComments**.

5.2 Uitwerking

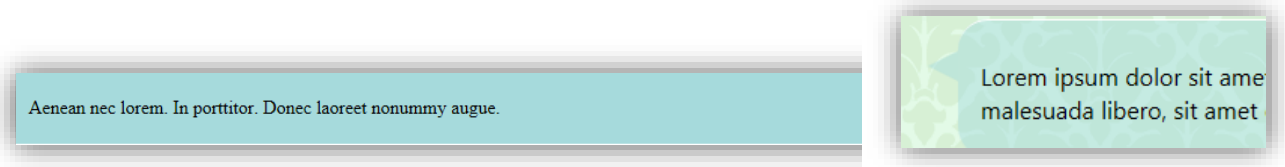
5.2.1 Background

Zorg ervoor dat de background.gif gebruikt wordt als achtergrond. Zorg dat er geen margin is maar wel een padding van 40px langs de 4 zijden.

Gebruik volgende fonts: "Segoe UI", Segoe, Calibri, Arial, sans-serif;

5.2.2 word-wrap

In de eerste stap zorg je ervoor dat lange woorden afgebroken worden en verder gaan op volgende lijn. Maak een stijl op **blockqoute** zodat je volgende weergave krijgt:



5.2.3 border-radius

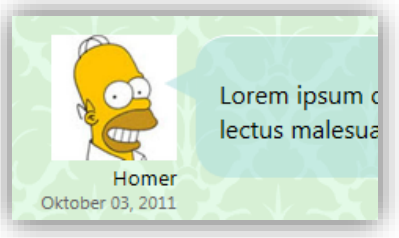
Pas de stijl die je zo juist in de blockquote hebt aangemaakt aan zodat deze ronde hoeken krijgt.

5.2.4 Tekstballon doorzichtig maken

Zorg ervoor dat het patroon van de background zichtbaar is doorheen de tekstballon. Om dit te doen moet je de achtergrond kleur van de tekstballon doorzichtig maken.

5.2.5 Driehoek aan tekstballon maken

Een tekstballon is pas een echte tekstballon wanneer je een driehoek of een verwijzing maakt naar bijvoorbeeld een persoon, in ons geval een image.



5.2.6 Zorg voor een hover effect op ons tekstballon

Om een hover te verkrijgen gaan we een schaduw om onze ballon plaatsen. Daarnaast gaan we ook onze ballon een klein beetje verhogen en klein beetje naar links plaatsen.

5.2.7 Zorg er nu voor dat de images mooi naast de tekstballonnen komen te staan

In eerste instantie zorgen we ervoor dat de cijfers die bij onze ordered list horen niet meer getoond worden. In volgende instantie schrijven we stijlcode die ervoor zorgt dat elke comment links uitgelijnd staat.

Doe hetzelfde voor de metadata die bij een comment hoort:

- 100px breed
- Het font is kleiner in de gewone tekst, dan in de tekstballon
- Metadata staat rechts uitgelijnd

Als laatste voorzie je een kleine rotatie op de image

6 Oefening 2: Media Queries

Voor deze oefening open we OefeningMediaQueries vanuit onze bronmateriaal map. Deze oefening is uit het boek Stunninng CSS genomen, dus zijn er geen credits verdiend door de docenten.

Bedoeling van deze oefening is dat we een webpagina gaan aanpassen op basis van de grootte van de browser of het toestel waarin de pagina bekeken wordt.

Open de pagina eens in een browser en verklein en vergroot het venster.

6.1 Stijl maken voor grote schermen

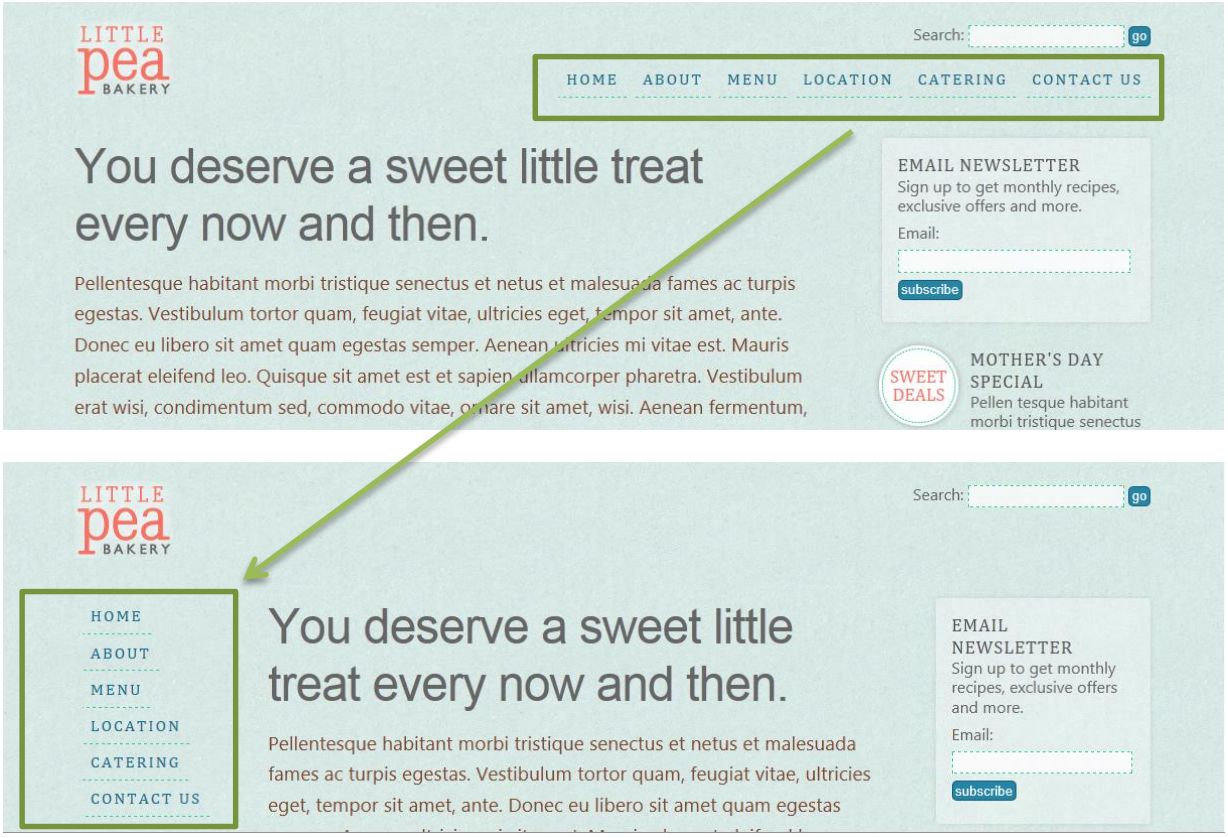
Voorzie **onderaan** de stijlen die je als bronmateriaal hebt meegekregen de mediaquery om schermen breder dan 1200px te targetten:

```
@media screen and (min-width: 1200px)
{

}
```

De reden waarom deze code onderaan moet staan is, omdat we de default stijlen willen overschrijven.

6.1.1 Verander de layout van 2 naar 3 kolommen



Pas het menubalk aan zodat deze een verticale wordt in plaats van een horizontale.

6.1.2 Multi-column tekst

In de volgende stap gaan we er voor zorgen dat de tekst in meerdere kolommen getoond worden. Dit kunnen we doen via de column-count property:

```
h1 + p
{
```

```
    -moz-column-count: 2;
    -moz-column-gap: 1.5em;
    -o-column-count: 2;
    -o-column-gap: 1.5em;
    -webkit-column-count: 2;
    -webkit-column-gap: 1.5em;
    column-count: 2;
    column-gap: 1.5em;
    color: #7F4627;
    text-shadow: -1px -1px 0 hsla(0,0%,100%,.6);
    font-size: 120%;
}
```

6.2 Stijl maken voor smalle schermen.

Voeg een media query toe om schermen kleiner dan 760 px een andere weergave te geven:

7 Oefening 3: Selectors

Ook deze oefening is een oefening uit het boek Stunning CSS. Open de OefeningenSelectors site vanuit ons bronmateriaal map.

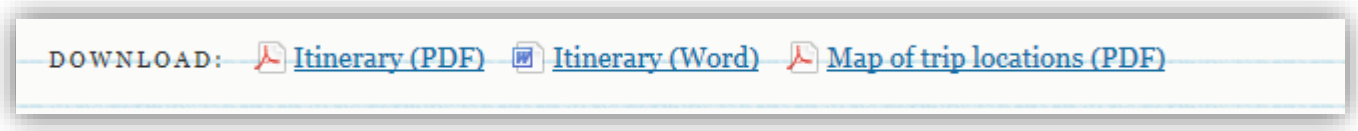
Start de site en bekijk wat we al reeds hebben.

7.1 Attribute selectors

In stap 1 gaan we elke link selecteren en wanneer die eindigt met volgende extensies, dan gaan we de corresponderende image ervoor tonen:

- Pdf

- Doc of docx
- Mov
- jpg



7.2 Selecteren op type

Selecteer de img elementen waarvan de source images uit de thumbnails map komen en zorg ervoor dat die links staan ipv rechts op de pagina:

```
img[src*=thumbnails]
{
    float: left;
    margin: 0 20px 10px 0;
}
```

Selecteer nu alle img's die uit de photos map komen en zorg dat ze er uit zien alsof het een polaroid image is: