

My own style

*Alle code hier besproken is op github beschikbaar:
<https://github.com/bazzel/de-zeepkamer>*

Voor deze opdracht is de markup (HTML) en styling (CSS) voor de home page gemaakt. Dit document beschrijft de werkwijze die hierbij is gehanteerd en de structuur die hiervoor is opgezet. Het eindresultaat is in onderstaande afbeelding te zien.



Voor het creëren van de HTML en CSS is gebruik gemaakt van [Middleman](#). Middleman is een tool die het mogelijk maakt om statische sites te genereren en maakt zelf weer gebruik van allerhande andere tools en handigheidjes.

Voor deze opdracht is met name de mogelijk om gebruik te kunnen maken van [Sass](#) voor het schrijven van CSS, [erb en templates](#) voor het creëren van HTML en de optie voor het genereren van CSS en HTML (build proces) erg handig. De uitwerkingen zijn, naast de geüploade bestanden bij de opdracht, ook terug te vinden op [github](#). In de volgende paragrafen zal steeds hiernaar worden verwezen.

Een klein, maar belangrijk detail: de teksten op de pagina zijn nietszeggende ('Lorem Ipsum') teksten en worden tijdens het renderen van de pagina gegenereerd. Een voordeel hiervan is dat het op deze manier eenvoudig is een pagina te voorzien van content en daarnaast is de lengte van de tekst variërend, waarmee snel is te zien of de styling schaalbaar is en zich hetzelfde gedraagt bij verschillende content.

Sass

Zonder reclame te maken voor Sass (iets wat voor open source op zich al vreemd is), moet gezegd worden dat, hiermee eenmaal kennis gemaakt te hebben, men nooit meer rechtstreeks CSS wil schrijven (tenzij er geen andere keuze is). Sass biedt 2 'talen': Sass en Scss.

Beide zijn een uitbreiding op CSS en biedt extra functionaliteit zoals het gebruik van variabelen, nesting, mixins, etc. Hierdoor kunnen bijvoorbeeld kleurcodes of lettertypes eenmalig worden gedefinieerd waarna deze door verschillende selectors kunnen worden gebruikt. Dit voorkomt het gebruik van dubbele attribute values en is het stukken eenvoudiger om tijdens het schrijven van CSS zaken uit te proberen.

In onderstaand (vereenvoudigd) voorbeeld wordt met Scss een aantal variabelen gedefinieerd voor het gebruik van tekst- en achtergrondkleuren van een navigatiemenu. Daaronder is de CSS zichtbaar die tijdens het build proces wordt gegenereerd:

```
// Scss:  
$menu-font-color      : white;  
$menu-item-bg         : hsl(42, 36%, 37%);  
$menu-item-hover-bg   : hsl(43, 54%, 27%);  
  
#menu {  
  ul {  
    li {  
      color: $menu-font-color;  
      a {  
        background: $menu-item-bg;  
        &:hover {  
          background: $menu-item-hover-bg;  
        }  
      }  
    }  
  }  
}  
  
// Css:  
#menu ul li {  
  color: white;  
}  
#menu ul li a {  
  background: #806d3c;  
}  
#menu ul li a:hover {  
  background: #6a5520;  
}
```

Opinionated markup

De ideale HTML pagina bevat uitsluitend semantische markup en wordt niet 'vervuld' met extra DIV's die nodig zijn voor het goed kunnen stylen van een pagina.

De praktijk is echter anders. Pagina's worden vaak gegeneerd d.m.v. een CMS en deze bepaalt voor een groot deel de markup.

Bij deze opdracht is er vanuit gegaan dat als CMS [Refinery](#) wordt gebruikt en is de bijbehorende markup als uitgangspunt genomen.

SMACSS

De CSS is in één bestand gedefinieerd. Je wilt er nl. voor zorgen dat een browser voor het renderen van een pagina zo weinig mogelijk requests naar een server hoeft uit te voeren. Het serveren van meerdere stylesheets vertraagt het tonen van de pagina en daarmee de user experience. Naast het serveren van CSS files zijn ook andere bestanden (scripts, images, etc.) van invloed, maar dat terzijde.

Omdat we gebruik maken van Sass, kunnen we voor het definiëren van de CSS wèl een modulaire opzet hanteren en deze over meerdere files verdelen, wat alles een stuk overzichtelijker en beter onderhoudbaar maakt. Het uiteindelijke build proces zorgt dat we met één enkele CSS file eindigen.

Voor deze modulaire opzet volgen we de principes van [SMACSS](#); SMACSS is een style guide voor het opzetten van CSS en schrijft bv. voor dat zaken als typografie, layout en elementen op een pagina in aparte files worden gedefinieerd.

Reset

Voor het stylen van de home page is gestart met het gebruik van een [reset.css](#). Hiermee worden de verschillen in styling tussen browsers, wanneer geen CSS wordt toegepast, rechtgetrokken.

Typografie

Er wordt gestart met de styling voor de typografie, zoals de opmaak van standaard elementen als paragrafen (P), headers (H1, H2, ...), lijsten (UL, OL, LI), enz.

Met sites als [<html>ipsum](#) en [Loripsum.net](#) kan een pagina eenvoudig worden voorzien van de benodigde markup.

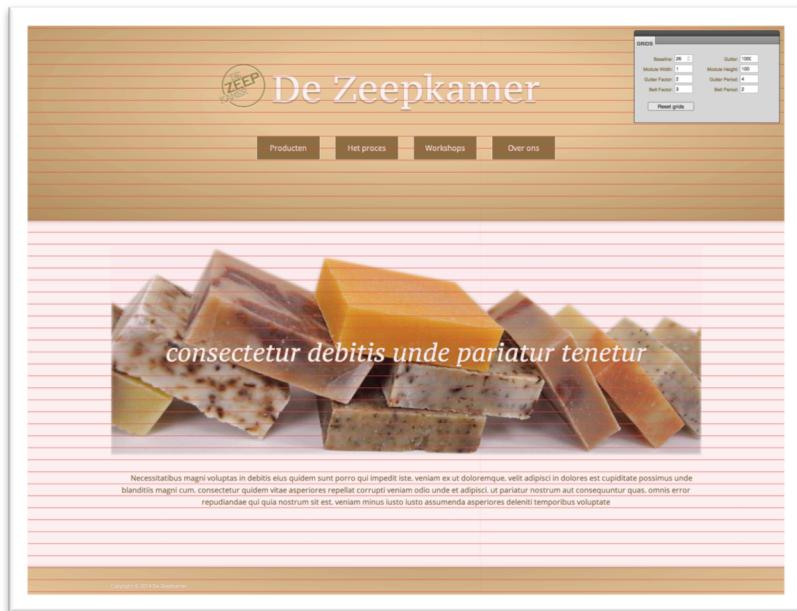
Zie ook de [markup](#) en [CSS](#) op github.

Vertical rhythm

Vertical rhythm stamt al uit de tijd van de boekdrukkunst en brengt meer evenwicht en rust in een pagina doordat elke tekstregel in een virtueel grid past waarbij de onderlinge afstand tussen de regels een vaste waarde (de regelhoogte) heeft.

Het gebruik van vertical rhythm is geen vereiste, maar het resultaat loont en het is een leuke uitdaging om dit in een pagina toe te passen.

[Compass](#) biedt hiervoor wat tooling en neemt het uitrekenen van de benodigde paddings, margins, line-heights, font-sizes en border-widths voor haar rekening. Onderstaande afbeelding toont dezelfde homepage, waarbij een verticaal grid van 26px zichtbaar is. Merk op dat tekstrgels steeds binnen dit grid passen en dat de verticale uitlijning van elementen zoals de navigatie, header en footer op het grid zijn afgestemd.



Layout

Met layout wordt de globale pagina indeling bedoeld, zoals de positie van de header, het navigatiemenu, de footer, wordt de content in meerdere kolommen getoond, enz. De layout zegt niets over andere stijlkenmerken zoals lettertypes, kleuren of regelhoeften.

De layout kan vanaf scratch worden opgezet of er kan gebruik worden gemaakt van een van de talloze hiervoor beschikbare frameworks. Voor deze opdracht heb ik gebruik gemaakt van [Susy](#), omdat deze geen eisen stelt aan de markup waardoor deze zoveel als mogelijk semantisch kan blijven.

De layout is eenvoudig: de header en footer beslaan de volledige breedte. Voor de daartussenliggende content wordt een kolom van maximaal 80em gehanteerd.

Zie ook de [layout](#) file op [github](#).

Modules en componenten

Elk deel op een pagina wat als zelfstandig element gezien kan worden, wordt als module of component beschouwd en de hiervoor benodigde CSS wordt in een eigen bestand gedefinieerd. Zo is de header een component, wat op zich weer componenten, zoals het navigatiemenu en de titel met logo, bevat.

Een eigenschap van een component en de hierbij behorende CSS, is dat het op elk deel van een pagina kan worden geplaatst, zonder dat hiervoor de CSS aangepast hoeft te worden.

De CSS voor de componenten is op [github](#) terug te vinden.

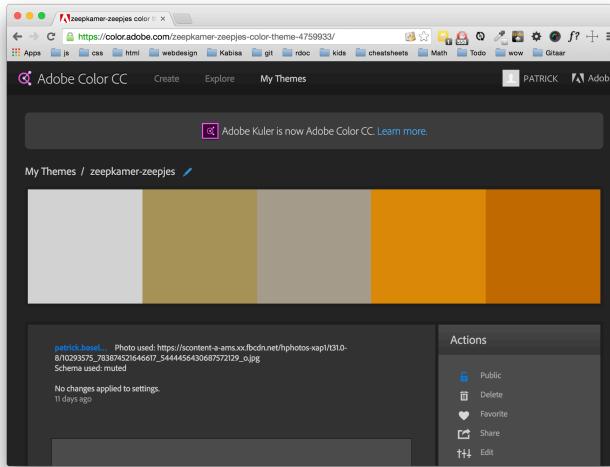
Thema

Naast de typografie, layout en het stijlen van modules adviseert SMACSS om een thema file te creëren waarin alle CSS regels staan die een bepaald thema beschrijven. Het is discutabel en aan de programmeur te bepalen welke regels

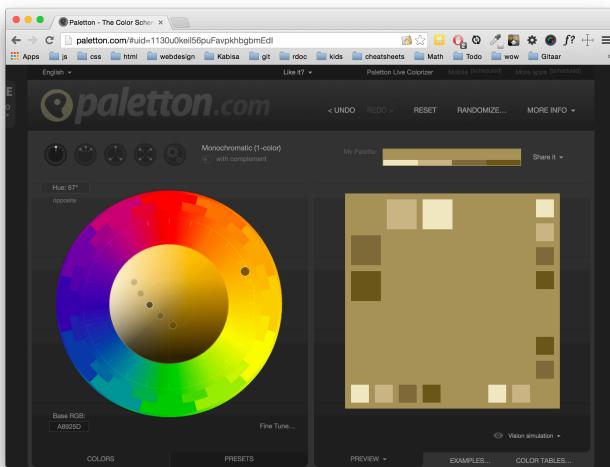
wel en niet onder een thema vallen, maar voorbeelden hiervan zijn bv. voor- en achtergrondskleuren, lettertypes, schaduw(-kleur), etc.

Kleurenschema

De aanwezigheid van de juiste kleuren (of juiste het ontbreken ervan) bepaalt voor een groot deel de uitstraling en het karakter van de site. Het samenstellen van het juiste kleurenpaljet is een vak apart, maar [Adobe Color CC](#) heeft me hierbij geholpen. Hier is het nl. mogelijk om aan de hand van bv. een foto (in dit geval een productfoto op de frontpage) een kleurenschema te genereren.



Het kleurenschema bevat meestal 5 verschillende kleuren en/of tinten. Dit kleurenschema is aanpasbaar door hier verschillende regels op toe te passen. Uit het gegenereerde schema heb ik 2 kleuren gekozen, waarvan vervolgens met [Palleton](#) verschillende tinten mee zijn gevormd.



Omdat het, zoals eerder gezegd, m.b.v. Sass mogelijk is variabelen te definiëren en het eveneens de mogelijkheid biedt te rekenen met kleuren, kan een kleurenschema uitgaande van een basiskleur (in dit geval #A8925D) worden berekend. Tinten van dezelfde kleur hebben nl. wanneer uitgedrukt in HSL i.p.v. RGB dezelfde (of nagenoeg dezelfde) hue (kleur) waarde.

In bovenstaande afbeelding kan de basiskleur worden uitgedrukt in $hsl(42, 30\%, 51\%)$. Een lichtere tint heeft de kleurcode $hsl(42, 59\%, 77\%)$, een donkerdere tint $hsl(42, 54\%, 27\%)$.

Dit biedt de mogelijkheid om de uitstraling van de site eenvoudig te wijzigen doordat uitsluitend de basiskleur gewijzigd hoeft te worden, wat het een stuk eenvoudiger maakt te experimenteren om tot het gewenste resultaat te komen.



Gebruikte links

Uitwerking opdracht	https://github.com/bazzel/de-zeepkamer	
Middleman	http://middlemanapp.com/	Middleman is a static site generator
Sass	http://sass-lang.com/	CSS extension language
Refinery CMS	http://refinerycms.com/	Ruby on Rails CMS
SMACSS	https://smacss.com/	A flexible guide to developing sites
Reset CSS	http://meyerweb.com/eric/tools/css/reset/	stylesheet to reduce browser inconsistencies
<html>ipsum	http://html-ipsum.com/	HTML fragmenten
Lorem ipsum.net	http://loripsum.net/	HTML fragmenten
Compass	http://compass-style.org/	CSS authoring framework
Susy	http://susy.oddbird.net/	CSS framework voor layout
Adobe Color CC	https://color.adobe.com	Color themes generator
Paletton	http://paletton.com	Color scheme designer