

Theming manual   
IP 3.4

Inhoudsopgave

1 the AIM themes 4

1.1 basetheme 4

1.2 the\_aim\_theme 4

1.3 the\_aim\_theme\_commerce 4

2 Gebruikte tools en technologieën 5

2.1 Grunt 5

2.2 SASS 8

2.3 Bourbon 13

2.4 Neat 14

2.5 Live reload 14

3 Responsive 14

3.1 Mobile first 14

3.2 Progressive enhancement 15

3.3 Adaptive images 15

4 Structuur van het theme 16

4.1 css 16

4.2 scss 16

4.3 svg 26

4.4 templates 26

5 Image processing 26

5.1 Drupal afbeelgingen 26

5.2 Theming afbeeldingen 27

6 Display suite 28

7 CKEditor 29

7.1 CKEditor styles 29

7.2 Inline images 29

7.3 CKEditor templates 30

8 Theming klasses 31

8.1 Beschikbare klasses 31

8.2 Gebruik van klasses 33

9 Verloop van een project 37

9.1 Lokale development opzetten 37

9.2 Bestudeer draaiboek en design 37

9.3 Project variabelen 38

9.4 Werk een eerste pagina uit. 38

9.5 Werk de overige pagina’s uit 38

9.6 Afwerking 38

# the AIM themes

Binnen the AIM hebben we een aantal themes ontwikkeld voor de uitwerking van onze websites. Deze themes blijven constant in ontwikkeling en er komen regelmatig nieuwe versies uit. Deze manual is van toepassing op de themes die zich in het Install Profile 3.4 bevinden.

## basetheme

Dit is het basistheme waarop de andere themes gebaseerd zijn. Het bevat geen styling, maar vooral configuratie. Zo hebben we in het basetheme gezorgd voor template suggesties, basis templates en gaan we hier vooral zorgen voor propere en overzichtelijke markup.

Je zal dit theme niet terugvinden in je themes folder. Over het algemeen zal je dit ook niet nodig hebben. Moest je dit toch nodig hebben om iets aan te passen of iets te bekijken, kopieer het basetheme dan vanuit de folder /profiles/theaim\_base/themes/ naar je eigen themes folder.

## the\_aim\_theme

Dit theme wordt gebruikt in onze Eco, Plus en Pro websites.

## the\_aim\_theme\_commerce

Dit theme wordt gebruikt in onze Eco, Plus en Pro webshops. Het bevat alle styling en configuratie van the\_aim\_theme, aangevuld met styling en configuratie specifiek voor webshops.

# Gebruikte tools en technologieën

Hier volgt een overzicht van tools, plugins en technologieën die gebruikt worden in onze themes.

## Grunt

GruntJS, gebouwd op [Node.js](http://nodejs.org/), is een taak-gebaseerde command-line tool waarmee je de development workflow kan versnellen. Dit door vooraf gedefinieerde taken uit te voeren, die automatisch worden uitgevoerd tijdens de ontwikkeling.

Grunt steunt op plugins om taken uit te voeren. Deze plugins worden regelmatig geüpdatet en er komen ook alsmaar nieuwe bij.

We kozen voor Grunt voor zijn snelheid van compileren en uitbreidbaarheid enerzijds, maar anderzijds zorgt het ervoor dat we binnen het team allemaal op een consistente manier werken.

Meer informatie over Grunt en de beschikbare Grunt plugins kan je vinden op <http://gruntjs.com/>.

### Grunt installeren

Binnen het team werken we met een vritualbox (FrontEndDeveloperMachine) waarop Grunt reeds geïnstaleerd is. Voor de volledigheid volgt hier toch de werkwijze voor het installeren van Grunt.

GruntJS is gebouwd op NodeJS, daarom moet dit eerst geïnstalleerd worden. Een eenvoudige manier hiervoor is met behulp van [Homebrew](http://brew.sh/).

Installeren homebrew

ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

Installeren NodeJS

brew install node

Installeren GruntJS

npm install -g grunt-cli

### Grunt gebruiken

Aangezien Grunt standaard voorgeconfigureerd in het theme aanwezig is, kan je met 1 commando alle nodige packages en plugins installeren. Ga hiervoor naar de root van de theme folder, waar het bestand package.json zich bevindt, en voer het volgende commando uit:

npm install

package.json

In dit bestand worden alle plugins gedefinieerd die gebruikt zullen worden in het project. Het commando npm install gaat deze plugins installeren en plaatst deze in de folder node\_modules.

**Let op:** Zorg ervoor dat je de folder node\_modules nooit meepusht naar de git repository.

Gruntfile.js

Dit bestand bevat volgende zaken

* Een “wrapper” functie
* Configuratie van de plugins
* Inladen van grunt plugins en taken
* Custom taken

Je zal deze 2 bestanden normaal gezien nooit moeten aanpassen of uitbreiden. Je kan er dus gewoon mee aan de slag.

### Grunt taken

Binnen onze themes hebben we 3 verschillende taken gedefinieerd die je kan gebruiken na het uitvoeren van de installatie (npm install).

grunt

Dit is het default commando. Voer dit commando uit tijdens de uitwerking/ontwikkeling van een project. Dit commando staat in voor volgende zaken:

* sass  
  scss bestanden zullen bij het opslaan ervan gecompileerd worden naar css bestanden.
* watch

Bij het opslaan van scss, js en tpl bestanden zal, na het compilen, een livereload plaatsvinden. De browser zal dus automatish de webpagina opnieuw inladen.

grunt svg

Voer dit commando uit wanneer er nieuwe afbeeldingen toegevoegd worden of bestaande afbeeldingen gewijzigd worden. Voor meer informatie over het gebruik en bewerking van SVG bestanden, zie het puntje ‘[Image processing’](#Images). Dit commando staat in voor volgende zaken:

* Sass

scss bestanden worden gecompileerd naar css.

* Includereplace

Afbeeldingen in de folder /svg zullen gekopieerd worden naar de folder /img/svg, en bepaalde zaken zullen vervangen worden. (Zie [‘Image processing’](#Images) voor meer info)

* svg2png

Alle svg bestanden in de folder /img/svg worden ook opgeslaan als png in de folder /img/png. Dit voor browsers die geen svg bestanden ondersteunen.

* sprite  
  Van alle png bestanden in de folder /img/png wordt een spritesheet gemaakt en in de folder /img geplaatst. Ook wordt het bestand \_sprites.scss geüpdated met de correcte css variabelen van de afbeeldingen.

grunt production

Voer dit commando uit wanneer je klaar bent met de uitwerking en voordat de website op staging of live geplaatst wordt.  
Dit commando staat in voor volgende zaken:

* Concat

Het bestand js/scripts.js wordt geconcatineerd en als js/build/production.js opgeslaan.

* uglify

Het bestand js/build/production.js wordt geminified en als js/build/production.min.js opgeslaan.

* tinypng

De spritesheet.png afbeelding wordt gecomprimeerd door middel van [tinypng](https://tinypng.com/).

* sass

scss bestanden worden gecompileerd naar css.

* watch

Bij het opslaan van scss, js en tpl bestanden zal, na het compilen, een livereload plaatsvinden. De browser zal dus automatish de webpagina opnieuw laden.

## SASS

Voor het schrijven van onze markup maken we gebruik van een CSS preprocessor, namelijk SASS. Een pre-processor is een stijltaal net zoals CSS maar zonder de limitaties van CSS. Een CSS pre-processor maakt het mogelijk om schone en overzichtelijke CSS code te schrijven en zorgt ervoor dat je variabelen, functies, berekeningen, overervingen en andere handige extra’s kan combineren met je reguliere CSS. Markup geschreven met een pre-processor is niet interpreteerbaar door browsers, daarom moet het worden vertaald worden naar CSS, in ons geval door Grunt.

Enkele voordelen van SASS worden hieronder toegelicht met voorbeeld:

Variabelen

Met behulp van variabelen kan je informatie opslaan die je vervolgens kan gebruiken over het hele project. Een groot voordeel hiervan is dat de waarde van een variabele maar op 1 plaats aangepast dient te worden.

Input:

$base-font-family: Arial, Sans-Serif;  
$text-color: #CCCCCC;

body {  
 font-family: $base-font-family;  
 color: $text-color;  
}

Output:

body {  
 font-family: Arial, Sans-Serif;  
 color: #CCCCCC;  
}

**CSS nesten**

In SASS is het mogelijk om CSS selectors te gaan nesten. Dit zorgt vaak voor meer overzichtelijke code. Let er wel op dat je zeker niet te diep gaat nesten. Dit resulteert immers in ellenlange selectors en styling die te specifiek op 1 element geplaatst wordt. Zorg er dus voor dat je je CSS niet complexer maakt door het nesten van selectors.

Input:

.main-menu {  
 float: left;  
 width: 100%;  
  
 ul {  
 display: block;  
 list-style: none;  
 margin: 0;  
 padding: 0;  
 }  
 li {  
 display: inline-block;  
 }  
 a {  
 display: block;  
 text-decoration: none;  
 }  
}

Output:

.main-menu {  
 float: left;  
 width: 100%;  
}  
.main-menu ul {  
 display: block;  
 list-style: none;  
 margin: 0;  
 padding: 0;  
}  
.main-menu li {  
 display: inline-block;  
}  
.main-menu a {  
 display: block;  
 text-decoration: none;  
}

Mixins

Met mixins kan je stijlen definiëren, al dan niet afhankelijk van argumenten, die herbruikt kunnen worden doorheen het project. Mixins kunnen volledige CSS-regels bevatten, alsook andere SASS code. Zie het als een functie die CSS-inhoud zou dumpen in plaats van een waarde retourneren. Mixins kunnen ervoor zorgen dat je bepaalde code niet alsmaar opnieuw dient te herhalen.

Input:

@mixin reset-list() {  
 list-style: none;  
 margin: 0;  
 padding: 0;  
}  
.main-menu {  
 ul {  
 @include reset-list;  
 }  
}

Output:

.main-menu ul {  
 list-style: none;  
 margin: 0;  
 padding: 0;  
}

Mixins kunnen ook werken met variabelen, al dan niet met een default value.

Input:

$black: #000000;  
$white: #FFFFFF;

@mixin rgba-bg($color:$black, $opacity:0.7) {  
 background: $color;  
 background: rgba(red($color), green($color), blue($color), $opacity);  
}

.news--item {  
 @include rgba-bg;  
}

.faq--item {  
 @include rgba-bg($white, 0.5);  
}

Output:

.news--item {  
 background: #000000;  
 background: rgba(0, 0, 0, 0.7);  
}

.faq--item {  
 background: #FFFFFF;  
 background: rgba(255, 255, 255, 0.5);  
}

Placeholders

Net als mixins, kan je ook met placeholders stijlen definiëren die herbruikt kunnen worden door het project. Placeholders kunnen echter niet werken met variabelen. Een ander verschil is dat je een placeholder niet met @include, maar met @extend oproept. Wanneer je @extend gebruikt wordt de CSS niet gedupliceerd, zoals bij @include, maar worden de selectors waar je dezelfde @extend gebruikt samengevoegd.

Placeholders schrijf je op dezelfde manier als een gewone klasse, met het enige verschil dat ze voorafgegaan worden door een % in plaats van een ..

Input:

%center {  
 display: block;  
 margin-left: auto;  
 margin-right: auto;  
}

.container {  
 @extend %center;  
}

.overview {  
 @extend %center;  
}

Output:

.container, .overview {  
 display: block;  
 margin-left: auto;  
 margin-right: auto;  
}

Extending classes

Niet alleen met placeholders, maar ook met gewone selectors kan je gebruik maken van @extend.

Input :

h3 {  
 font-weight: bold;  
}

p.title {  
 @extend h3;  
}

Output:

h3, p.title {  
 font-weight: bold;  
}

Referencing Parent Selectors

Het ampersand teken & wordt gebruikt om de parent van de huidige selector uit te breiden.

Input:

a {  
 text-decoration: none;  
  
 &:hover, &.active {  
 text-decoration: underline;  
 }  
}

Output:

a {  
 text-decoration: none;  
}  
a:hover, a.active {  
 text-decoration: underline;  
}

Commentaren

Commentaren in SASS kunnen vooraf gegaan worden door //. Het voordeel hiervan is dat deze commentaar niet in het gecompileerde CSS bestand getoond wordt.

Meer informatie en voorbeelden over SASS vind je op <http://sass-lang.com/>.

## Bourbon

Bourbon is een eenvoudige, lightweight en uitgebreide library van SASS mixins die heel eenvoudig en makkelijk te gebruiken is.

Voor een uitgebreid overzicht van alle mixins, functies en add-ons, neem een kijkje op <http://bourbon.io/docs/>.

## Neat

Neat is een lightweight, semantic grid-system dat naadloos samenwerkt met Bourbon en SASS.

Meer informatie en voorbeelden vind je op <http://neat.bourbon.io/>

## Live reload

Zoals eerder reeds aangehaald, maken we tijdens de ontwikkeling van websites gebruik van Live reload. Wanneer Grunt aan het draaien is zal bij het opslaan van scss, js en tpl bestanden, na het compilen, een livereload plaatsvinden. De browser zal dus automatish de webpagina opnieuw inladen.

Live reload zal enkel plaatsvinden wanneer “the AIM environmental variable” op “Development” staat. Deze setting kan aangepast worden op “the AIM general settings page” (/admin/config/the-aim/settings). Het is immers niet nodig dat dit op staging of live sites nog ingeschakeld staat.

# Responsive

Bijna alle website die we ontwikkelen binnen the AIM zijn responsive websites. Daarom zijn onze themes standaard reeds responsive aan de hand van media queries.

## Mobile first

Binnen the AIM ontwikkelen we alle responsive websites volgens het “**Mobile first**” principe. Dit betekent dat we zowel bij het prototype, het design als bij de uitwerking van een responsive website starten bij de mobiele versie van een pagina. Om zo, per breakpoint, op te bouwen naar de desktop versie.

De hoofdreden hierachter is dat, als je jezelf eerst beperkt tot mobiel, je verplicht bent om te beslissen wat echt belangrijk is en waar je de focus op wilt. Hierdoor moet je beslissingen nemen die je nooit moet nemen of niet bij stilstaat wanneer je eerst de desktop versie uitwerkt.

Een andere belangrijke reden, vooral dan bij de uitwerking van een website, is dat er op mobiel enkel de hoogst nodige css, afbeeldingen, … ingeladen worden, speciaal gericht op mobiele toestellen. Wanneer we verder gaan naar grotere schermresoluties worden er pas grotere afbeeldingen, extra css en overschrijvingen ingeladen. Wat de laadtijd op mobiele toestellen alleen maar ten goede komt.

## Progressive enhancement

Binnen the Aim maken we gebruik van **progressive enhancement.**

Progressive enhancement is een methode om scripts en CSS in webpagina's te verwerken. Hierbij wordt de webpagina aanvankelijk ontworpen voor de minst geavanceerde browser, waarna de meer geavanceerde functionaliteit er 'in lagen' overheen wordt gelegd.

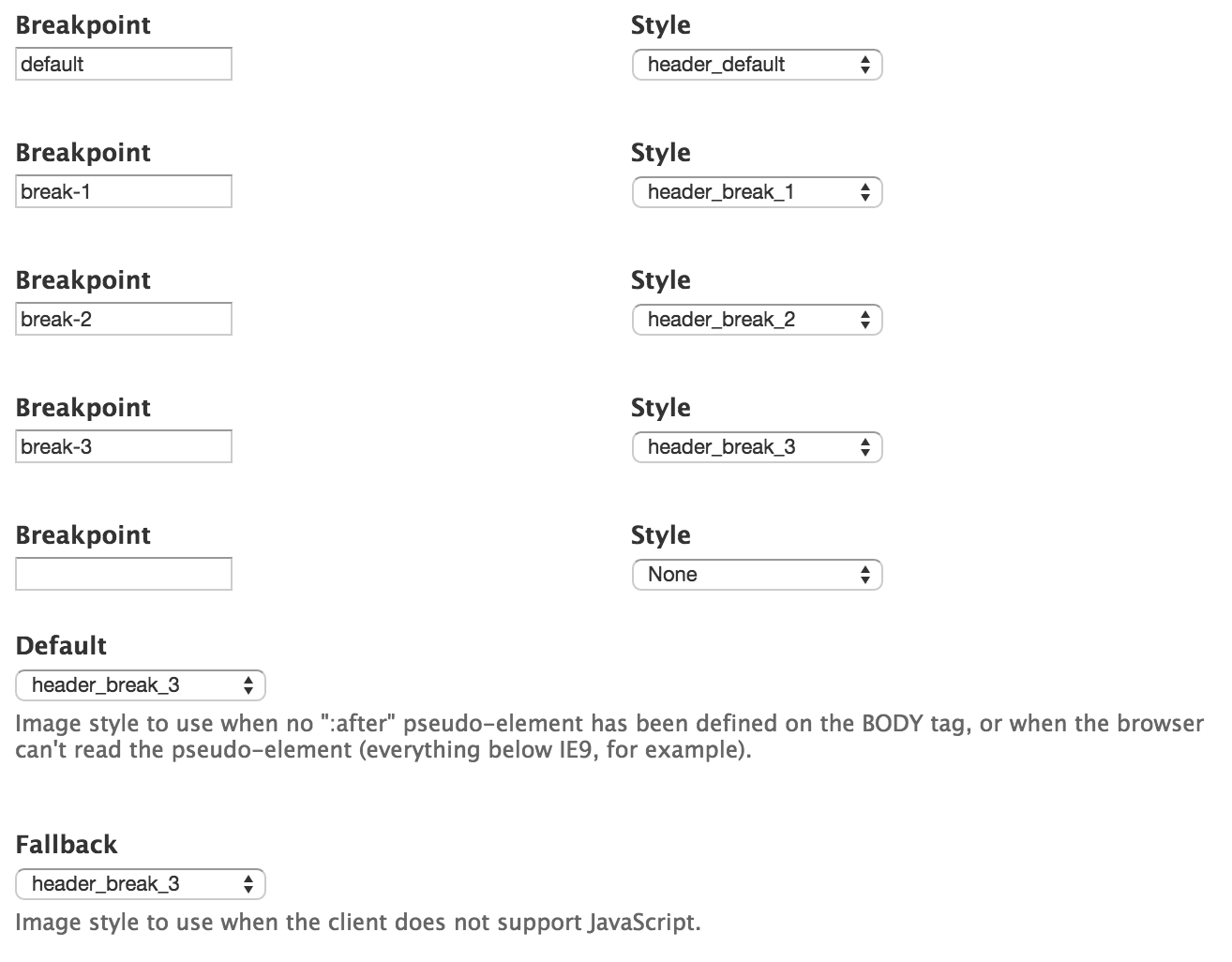
Uitgangspunt hierbij is dat minder geavanceerde browsers (denk aan een oude versie of bijvoorbeeld screenreaders) steeds terug kunnen vallen op een onderliggende laag wanneer zij functionaliteit tegenkomen die ze niet kunnen verwerken. (Zijnde bewegingen, schaduwen, transparantie, kleurverloop, etc.)

Naar **responsive design** toe zorgen we ervoor dat alles functioneert in de verschillende browsers. Kleine afwijkingen van het design zijn echter steeds mogelijk afhankelijk van de resolutie en het soort browser.

De browsers die we standaard ondersteunen zijn IE8, IE9, IE10, IE11, Mozilla firefox, Google Chrome, Safari en Opera.

## Adaptive images

Voor afbeeldingen binnen Drupal maken we gebruik van adaptive images. Het is aan te raden om volgende module hiervoor te gebruiken: <https://bitbucket.org/theaim/the_aim_cs_adaptive_image/>. Dit is een module, aangepast door the AIM als uitbreiding op <https://www.drupal.org/project/cs_adaptive_image>, wat ons toelaat om per breakpoint andere afbeeldingsprofielen te gaan gebruiken. Door de aanpassingen kunnen we gewoon de naam van ons breakpoint gebruiken in plaats van waarden in pixels.



# Structuur van het theme

We streven naar een vaste en consistente structuur binnen het theme. Dit zorgt ervoor dat iedereen op dezelfde manier werkt en iemand anders ook vlot aan jou project kan verder werken indien nodig. Hier wordt de structuur van het theme toegelicht.

## css

In deze folder zullen onze gegenereerde css bestanden, na compilatie van de scss bestanden, terecht komen. In deze folder zullen we dus nooit rechtstreeks werken.

## scss

In deze folder worden alle scss bestanden geplaatst. We werken hier met een vaste structuur. Hieronder een verduidelijking van de verschillende bestanden met hun inhoud.

### style.scss

Dit is het belangrijkste van de scss bestanden. Dit bestand zal gecompileerd worden naar style.css, dat overal ingeladen zal worden in de front-end van de website. In dit bestand worden de nodige libraries en verschillende scss bestanden ingeladen. Dit geeft ons de mogelijkheid om aparte bestanden te maken, die styling bevatten voor specifieke doeleinden, en toch te compileren naar 1 css bestand. Zo behouden we een duidelijk overzicht, heeft styling zijn eigen plaats, en moet er maar 1 request gedaan worden naar een css bestand.

Hier een overzicht van de belangrijkste bestanden die ingeladen worden en extra verduidelijking nodig hebben.

\_variables.scss

Zoals de naam al zegt, bevat dit bestand alle variabelen die we zullen gebruiken in het project.

**Font variabelen**

Hier laden we onze verschillende lettertypes in, die we nodig hebben in het project. In dit bestand staat ook een voorbeeld in commentaar om een nieuw lettertype te definiëren.

* $base-font-family: definieer hier het basis lettertype. Dit is het lettertype dat gebruikt zal worden voor alle lopende teksten.  
  vb: $base-font-family: $abel;
* $base-font-size: definieer hier de basis lettergrootte in pixels. Deze zal omgevormd worden naar em waarden.
* $base-line-height: definieer hier de basis lijnhoogte in pixels. Deze zal omgevormd worden naar em waarden.
* $em-base & $em-base-grid: deze variabelen zullen gebruikt worden om een correcte em berekening te doen. Opgelet: pas deze variabelen nooit aan, anders zal er een foute berekening gebeuren, en zullen lettergroottes en containerbreedtes niet correct zijn.

**Fixed grid variabelen**

Hier is 1 variabele aanwezig, namelijk **$fixed-container**. Standaard staat deze op **true**, wat ervoor zorgt dat de containers per breakpoint een vaste breedte hebben, namelijk de gedefinieerde breedte van het breakpoint zelf. Opteer je voor een fluid container? Zet de waarde van deze variabele op **false**. Dit zal ervoor zorgen dat de container altijd de breedte heeft van het scherm, behalve voor het laatste breakpoint. In toekomstige versies van het install profile zal deze variabele standaard op **false** staan, daar we liever opteren voor een fluid layout.

**Breakpoint variabelen**

Hier worden onze breakpoints gedefinieerd. Standaard zijn er 3 breakpoints aanwezig. Per breakpoint wordt de breedte (in pixels) en het aantal kolommen (voor onze grid) bepaald. Hier kan je, indien gewenst ook breakpoints toevoegen. Bijvoorbeeld:

//Widths per breakpoint  
break-default-w: 320;  
break-1-w: 480;  
$break-2-w: 768;  
$break-3-w: 960;  
$break-4-w: 1200;  
//Columns per breakpoint  
$break-default-c: 4;  
$break-1-c: 6;  
$break-2-c: 9;  
$break-3-c: 12;  
$break-4-c: 15;  
//Breakpoint rules  
$break-1: new-breakpoint(min-width em($break-1-w, $em-base-grid) $break-1-c);  
$break-2: new-breakpoint(min-width em($break-2-w, $em-base-grid) $break-2-c);  
$break-3: new-breakpoint(min-width em($break-3-w, $em-base-grid) $break-3-c);  
$break-4: new-breakpoint(min-width em($break-4-w, $em-base-grid) $break-4-c);

In het bovenstaande voorbeeld werd een nieuw breakpoint (**$break-4**) toegevoegd met een breedte van 1200px en 15 kolommen. Eens dit gedefinieerd is, kan je in andere bestanden styling toevoegen voor dit breakpoint.

In toekomstige versies van het install profile zal het breakpoint vanuit het voorbeeld hierboven standaard aanwezig zijn.

**Kolom instellingen**

Hier worden de instellingen voor de kolommen van het grid system gedaan. De enige variabele die je hier eventueel zal moeten veranderen is **$max-width.** Wanneer je, zoals in bovenstaand voorbeeld, een extra, groter breakpoint hebt aangemaakt, zal de maximum breedte van de website aangepast moeten worden naar dit breakpoint.

$grid-columns: $break-default-c;  
$column: (em($break-default-w/$break-default-c)/4)\*3;  
$gutter: em($break-default-w/$break-default-c)/4;  
$max-width: em($break-4-w);  
@if $fixed-container {  
 $max-width: em($break-default-w);  
}

**Grid variabelen**

Hier staan een aantal variabelen die een hulp kunnen zijn tijdens het themen.

* $show-firebug-lite: Toont de firebug lite link. Wat het debuggen op mobiele devices mogelijk maakt.
* $show-breakpoint-label: Toont rechts bovenaan een label met de naam van het breakpoint. Handig om te weten in welk breakpoint je je bevindt.
* $show-grid-background: Visualiseert het grid op de achtergrond.

**Kleur variabelen**

Ook kleur variabelen, voor alle kleuren gebruikt in een project, worden hier gedefinieerd.

components/\_neat\_overrides.scss

In dit bestand zal je nooit werken, en mag je ook nooit aanpassen, maar bevat enkele custom functies en mixins die bepaalde neat functies zullen overschrijven.

**@include span-columns**

Met deze mixin kunnen gaan we een element een bepaalde breedte geven, op basis van ons grid systeem. Enkele voorbeelden:

🡪 De social media links moeten 2 kolommen innemen:

.social {  
 @include span-columns(2);  
}

🡪 De social media links moeten 2 van 4 kolommen innemen:

.social {  
 @include span-columns(2 of 4);  
}

🡪 Het zoekvak moet 3 kolommen innemen, maar aan de rechterkant geplaatst worden (float: right)

.searchbox {  
 @include span-columns(2 omega);  
}

**@include omega**

Met deze mixin kunnen we, van elementen gestijld met de **@include span-columns** mixin, de **float** omkeren en de zij margin op 0 plaatsen.

Een voorbeeld ter verduidelijking:

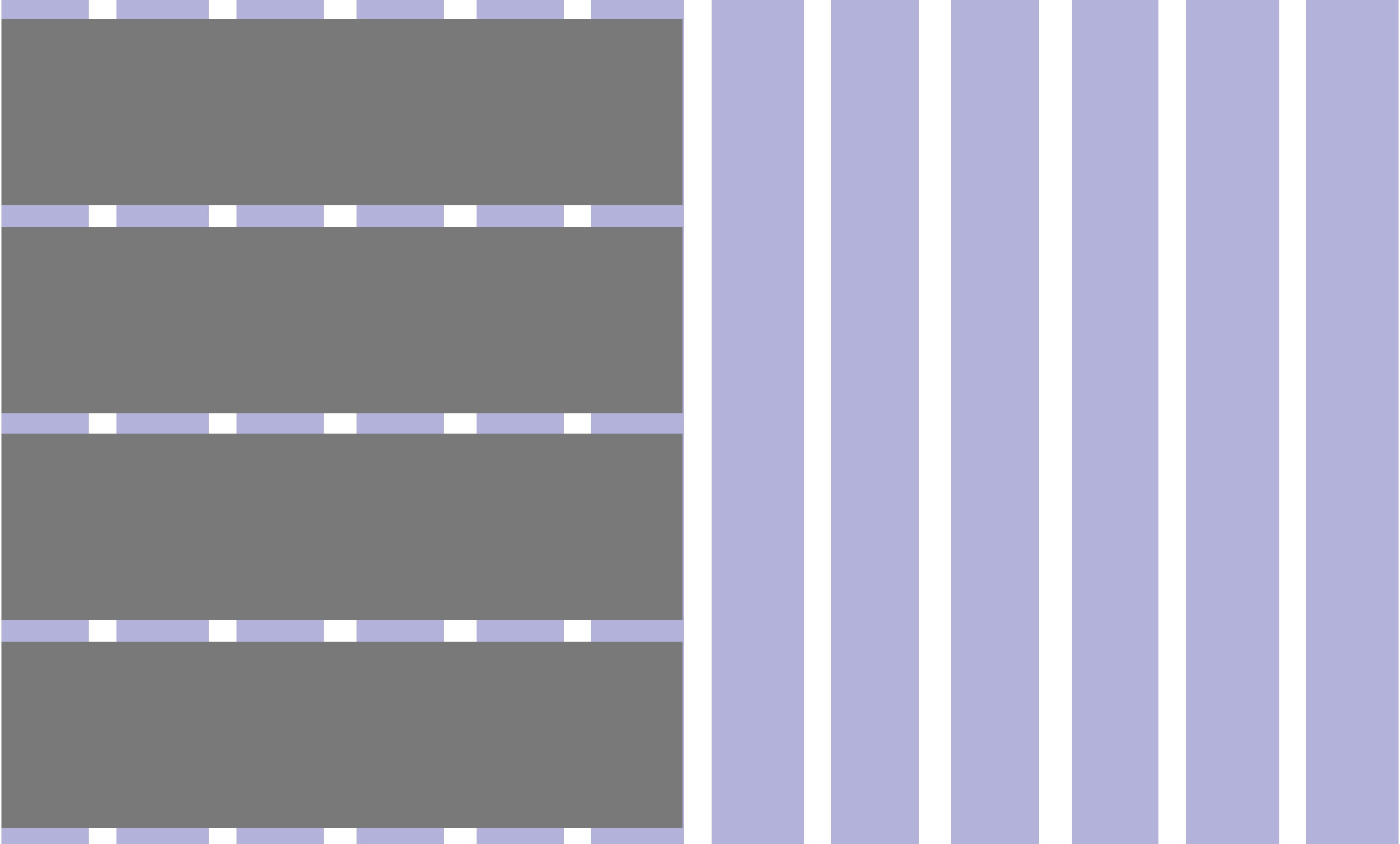
🡪 We hebben een overzicht van items die telkens de helft van de breedte mogen innemen en moet er als volgt uitzien:



Wanneer we het volgende doen:

.item{  
 @include span-columns(6);  
}

Zullen alle items een margin-right meekrijgen, met als gevolg dat items onder elkaar zullen staan:



Om dit op te lossen maken we gebruik van de mixin **@include omega**.

.item{  
 @include span-columns(6);  
  
 &.even {  
 @include omega;  
 }  
}

Bovenstaande code zorgt ervoor dat items met de klasse **.even** een **float: right;** en **margin-right: 0;** meekrijgen.

components/\_mixins.scss

Dit bestand bevat custom mixins. Wanneer je een eigen mixin wenst te schrijven, voeg je deze toe aan dit bestand. Er zijn hier reeds een aantal custom mixins standaard aanwezig:

**@include reset-list;**

Mixin om de css van een <ul> of <ol> te resetten.

Input:

.main-menu {  
 @include reset-list;  
}

Output:

.main-menu {  
 list-style: none;  
 margin: 0;  
 padding: 0;  
}

**@include opacity(opacity);**

Deze mixin zorgt ervoor dat opacity ook in IE8 werkt.

Input:

.news-item {  
 @include opacity(0.8);  
}

Output:

.news-item {  
 filter: progid:DXImageTransform.Microsoft.Alpha(Opacity=50);  
 opacity: 0.5;  
}

**@include rgba-bg(color, opacity);**

Deze mixin biedt een fallback voor IE8 bij transparante achtergronden;

Input:

.news-item {  
 @include rgba($white, 0.8);  
}

Output:

.news-item {  
 background: #FFFFFF;  
 background: rgba(255,255,255,0.8);  
}

**@include font(font-size);**

Dit is een custom mixin om een font-size in pixels om te zetten in em waarden.

**@include hyphens;**

Mixin te gebruiken voor hyphenation.

Input:

h1 {  
 @include hyphens;  
}

Output:

h1 {  
 -webkit-hyphens: auto;  
 -moz-hyphens: auto;  
 hyphens: auto;  
}

components/\_placeholders.scss

Dit bestand bevat custom placeholders. Wanneer je een eigen placeholder wenst te schrijven, voeg je deze toe aan dit bestand.

\_sprites.scss

Dit bestand wordt automatisch gegenereerd na het uitvoeren van het commando grunt svg en bevat alle informatie voor het correct weergeven van de spritesheet.

\_svg.scss

Dit bestand bevat de mixin voor het gebruik van svg afbeeldingen alsook de paden naar de locaties van de afbeeldingen. Meer info over dit bestand vind je onder het puntje ‘[Image processing](#Images)’

\_base\_include.scss

In dit bestand worden alle markup geplaatst voor basis tekststijlen zoals paragrafen, headings, links, lijsten, ….

\_forms\_include.scss

Hier wordt alle markup voor formulieren geplaatst.

plugins/\_colorbox.scss

Dit bestand wordt gebruikt om de colorbox popup op te maken. De algemene markup bovenaan dit bestand laat je best zo staan, aangezien dit voor elke website hetzelfde is. Onderaan vind je een aantal selectors voor de knoppen en achtergrondkleuren die je kan aanpassen naar de huisstijl van de website.

plugins/\_flexslider.scss

Dit bestand bevat de algemene styling voor de flexslider plugin.

plugins/\_checkout.scss

Dit bestand komt enkel voor in the\_aim\_theme\_commerce en bevat alle styling voor het checkout proces.

plugins/\_user.scss

Hier wordt alle styling verzameld wat betreft ingelogde gebruikers (login formulieren, registratie, …)

\_structure.scss

Dit bestand is eigenlijk de kapstok van onze website. Hierin wordt de styling van de algemene structuur van de website in opgenomen (containers, header, regions, footer). Gelieve dit ook zo te houden en hier enkel styling voor de hoofdstructuur van de website in te plaatsen.

\_entities.scss

In dit bestand wordt alle styling met betrekking tot entities in opgenomen (nodes, taxonomy, custom entities, view modes).

\_blocks.scss

Hierin worden alle blokken gestijld. Het gaat vooral om drupal blocks en the AIM blocks die hierin worden opgenomen. **Let op:** wanneer je een blok hebt met een overzicht van nodes of taxonomy, opteer er dan voor om deze styling in het bestand \_entities.scss te plaatsen.

\_menus.scss

In dit bestand wordt alle styling voor menus geplaatst.

Er zijn hier enkele belangrijke mixins die gebruikt kunnen worden. Deze mixins zijn terug te vinden in het bestand \_plugins/\_type\_menus.scss.

**@include add-menu(type, snelheid, hoogte)**

Met deze mixin kan je de styling van een mobiel menu (uitklapbaar) toevoegen. Voorlopig is er als type enkel het ‘burger’ menu beschikbaar, maar dit zal in de toekomst nog uitgebreid worden.

**@include rm-menu(type)**

Met deze mixin kan je de styling van het mobiel menu (in een hoger breakpoint) terug verwijderen, waardoor het een normaal horizontaal menu wordt voor op desktop.

\_admin.scss

In dit bestand wordt alle admin styling geplaatst. Het gaat hier over het tab menu om de content aan te passen, drupal messages en de contextual links van een block.

### svg.scss

Dit bestand wordt gebruikt voor de opmaak van svg afbeeldingen. Hier kan je de fills en strokes voor de svg’s aanpassen. Dit bestand wordt gecompileerd naar svg.css, dat aangesproken zal worden in onze SVG bestanden.

### stylguide.scss

In dit bestand worden alle basis markup ingeladen uit \_base\_include.scss en \_forms\_include.scss. Dit bestand kan gebruikt worden om in te laden in de CKEditor of in iframes gebruikt in de website.

## svg

In deze folder worden alle svg afbeeldingen geplaatst, die vervolgens verwerkt zullen worden.

## templates

In deze folder worden alle drupal templates verzameld. Enkel de basis templates worden hier in de root geplaatst (page.tpl.php, node.tpl.php, taxonomy-term.tpl.php). De meer specifieke templates worden telkens in duidelijk benaamde subfolders geplaatst. Dit zorgt ervoor dat het een overzichtelijk geheel blijft, en dat iedereen gebruikte templates snel kan terugvinden.

# Image processing

Hier wordt beschreven hoe we omgaan met afbeeldingen in onze websites.

## Drupal afbeelgingen

Dit punt behandeld afbeeldingen die opgeladen worden via de Drupal backoffice. Een aantal zaken waar we altijd op letten:

* Drupal image field settings  
  Voor afbeeldingvelden binnen Drupal stellen we altijd een minimum- en maximum grootte in. Dit zorgt ervoor dat er nooit te grote of te kleine afbeeldingen op de server opgeladen kunnen worden.  
  Ook gaan we altijd een ‘File directory’ instellen. Dit zorgt ervoor dat opgeladen afbeeldingen nooit rechtstreeks in de ‘Files’ folder komen. Zo blijft deze folder altijd heel overzichtelijk.
* Bestandsgrootte  
  Let er altijd op dat afbeeldingen die je oplaad via de backoffice niet te groot zijn. Opteer ervoor om afbeeldingen alvorens het opladen te comprimeren met tools zoals <http://www.jpegmini.com/> of <https://tinypng.com/>.
* Adaptive images

Kies ervoor, indien mogelijk en opportuun, om per breakpoint een apart afbeeldingsprofiel te gebruiken. Dit vorkomt dat er niet nodeloos te grote afbeeldingen worden ingeladen op mobiele devices.

## Theming afbeeldingen

Dit punt behandeld afbeeldingen die we gebruiken in ons theme.

### SVG afbeeldingen

We opteren ervoor zoveel mogelijk gebruik te maken van SVG afbeeldingen. Deze worden aangeleverd door de designer van dienst. Indien SVG bestanden niet aanwezig zijn voor een project, vraag deze dan op aan de designer.

We kiezen voor SVG afbeeldingen omdat de bestandsgrootte van deze bestanden doorgaans kleiner is, ze geschaald kunnen worden zonder kwaliteitsverlies en omdat deze er goed uitzien op retina schermen.

Hier volgt de werkwijze om een SVG afbeelding toe te voegen.

* Plaats de SVG in de folder /svg
* Geef de polygon, path, group, … in het bestand een ID, zodat je deze kan aanspreken in CSS.
* Voeg een link toe in het bestand naar ons CSS bestand. Doe dit onder de <svg> tag, en boven de polygon, path, group, …

<style type="text/css" media="screen">  
 @@include('../css/svg.css')  
</style>

* Voeg de variabele en het pad naar de svg toe in het bestand scss/\_svg.scss
* Variabele  
  Voeg de variabele toe aan de $defined-svgs array. De variabele naam is gelijk aan de bestandsnaam, voorafgegaan door $.
* Pad

Voeg het pad naar de SVG toe aan de $svg-paths array. Dit is het pad naar de bewerkte SVG, dit wil zeggen in de folder /img/svg in plaats van /svg.

Let op: de positie van de variabele en het pad moet overeen komen.   
Tip: Wanneer je een alfabetische volgorde hanteerd, is dit altijd correct.

* Voeg de nodige css toe aan scss/\_svg.scss dit. Je kan een fill of eventueel een stroke toevoegen.
* Voer het commando grunt svg uit.   
  Dit zorgt ervoor dat afbeeldingen in de folder /svg gekopieerd zullen worden naar de folder /img/svg en het bestand svg.css ingeladen wordt. Vervolgens worden alle svg bestanden in de folder /img/svg opgeslaan als png in de folder /img/png. Dit voor browsers die geen svg bestanden ondersteunen. Ten slotte worden er van alle png bestanden in de folder /img/png een spritesheet gemaakt en in de folder /img geplaatst. Ook wordt het bestand \_sprites.scss geüpdated met de correcte css variabelen.
* Vanaf nu kan je de SVG gebruiken in je styling door middel van de mixin @include svg($variabele);. Deze mixin zorgt voor een png fallback die gebruikt zal worden door browsers die geen svg bestanden ondersteunen.

# Display suite

Je zal merken dat de module ‘Display suite’ standaard aanwezig is. In toekomstige versies van het install profile zal deze niet meer aanwezig zijn. Gelieve nu al in de mate van het mogelijke zoveel mogelijk gebruik te maken van template files en preprocess functies in plaats van de Display suite functionaliteit. Deze doet soms meer fout dan goed, en dit geeft je meer controle en overzicht over wat je net aan het doen bent.

# CKEditor

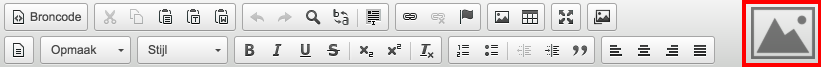
We maken in de drupal backoffice gebruik van CKEditor. Hieronder worden enkele extra features besproken waarvan we gebruik maken.

## CKEditor styles

In de root van de theme folder is er een bestand **ckeditor.styles.js** aanwezig. Hierin kan je zelf stijlen definieren zoals buttons, speciale tussentitels, … die je vevolgens in de editor kan gebruiken.

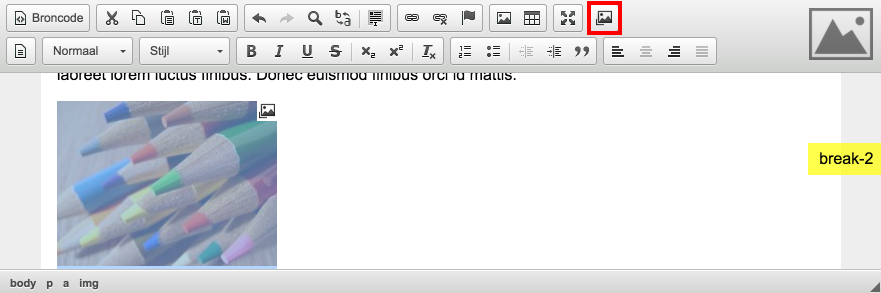
## Inline images

Je zal zien dat voor alle content types die gebruik maken van het veld *field\_images* (Basic Page, News) er een extra knop aanwezig is in de editor, namelijk een afbeeldingsknop.



Wanneer we op deze knop klikken, schuift het afbeelding veld uit langs de zijkant, en kunnen we hier afbeeldingen opladen. Eens we een afbeelding opgeladen hebben, verschijnt er een insert knop (van insert module), waarmee we de afbeelding tussen de tekst kunnen plaatsen.

Wanneer we vervolgens in de editor de afbeelding selecteren, kunnen we op de knop ‘Fotogalerij’ klikken, om ervoor te zorgen dat de afbeelding in een colorbox popup geopend kan worden.



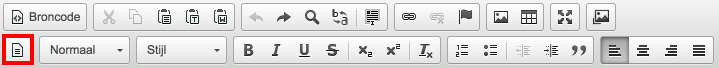
De afbeelding in de editor krijg een extra icoon, om aan te geven dat ‘Fotogalerij’ is toegepast op de afbeelding.

Met de stijlen ‘Image on Left’ en ‘Image on Right’ kan je de afbeelding links of rechts van een tekstblok plaatsen.

## CKEditor templates

Er is ook de mogelijkheid ingebouwd om CKEditor templates aan te maken. Hiermee kan je verschilende types van inhoud aanmaken, zonder dat je hiervoor een apart content type dient aan te maken. Ga hiervoor als volgt te werk:

* Voeg in de folder /templates/ckeditor een nieuw html bestand toe met een duidelijke naamgeving.
* Op de eerste regel van dit bestand wordt de naam geplaatst. Deze naam zal zichtbaar zijn in de editor.
* Op de tweede regel van dit bestand wordt de beschrijving geplaatst. Deze beschrijving zal zichtbaar zijn in de editor.
* Voeg vanaf de derde regel de gewenste html toe.
* Eindig het bestand met een lege paragraaf <p>&nbsp;</p>. Dit zorg ervoor dat je onder de template eenvoudig content kan toevoegen in de editor.
* Voeg de nodige styling toe in het daarvoor geschikte scss bestand.
* De nieuwe template is nu beschikbaar in de editor, onder de knop ‘Sjablonen’.



# Theming klasses

## Beschikbare klasses

Hier volgt een overzicht van klasses die standaard in het theme aanwezig zijn en waarvan je gebruik kan maken. Voor deze klasses is er reeds standaard styling aanwezig in het bestand dat tussen haakjes vermeld wordt.

### .container (\_structure.scss)

Deze klasse wordt gebruikt als containerklasse binnen een region. Dit zorgt ervoor dat de region de volledige breedte van het scherm heeft, en de container maximum de breedte van het breakpoint.

<header>  
 <div class=”container”>  
 …  
 </div>  
</header>

### .main-menu (\_menus.scss)

Deze klasse wordt gebruikt voor het hoofdmenu van de website, dat standaard aanwezig is na installatie.

### .secondary-menu (\_menus.scss)

Deze klasse wordt gebruikt voor het secundair menu van de website, dat standaard aanwezig is na installatie.

### .sub-menu (\_menus.scss)

Deze klasse wordt gebruikt voor een submenu, dat zich in een sidebar bevindt.

### .footer-menu (\_menus.scss)

Deze klasse wordt gebruikt voor het footer menu, dat standaard aanwezig is na installatie.

### .more (\_entities.scss)

Deze klasse kan op een paragraaf geplaatst worden, waarin zich een “Lees meer” link bevindt. Bijvoorbeeld bij het nieuws overzicht.

### .overview (\_entities.scss)

Gebruik deze klasse voor overzichten waar alle items onder elkaar weergegeven moeten worden. Ze hebben elk een breedte van 100%.

### .responsive (\_entities.scss)

Gebruik deze klasse op een overzicht van list items. Deze worden standaard op mobiel met 2, op tablet met 3 en op desktop met 4 langs elkaar getoond.

### .imgs-block (\_entities.scss)

Gebruik deze klasse op een afbeeldingsveld mer meerdere waarden. Het zal de eerste afbeelding tonen als een groot beeld, en de volgende afbeeldingen als thumbnails onder de grote afbeelding. Zorg er wel voor dat de eerste afbeelding gebruik maakt van een ander afbeeldingsprofiel.

### .sharelinks (\_entities.scss)

Deze klasse wordt automatisch gebruikt op de wrapper van de sharelinks, die per content type ingeschakeld kunnen worden op /admin/config/the-aim/social.

### .language (\_blocks.scss)

Deze klasse wordt standaard gebruikt op de taalkeuze block.

### .shopping-cart (\_blocks.scss)

Deze klasse wordt standaard gebruikt op het winkelmandje in shops.

### .searchbox (\_blocks.scss)

Deze klasse wordt standaard gebruikt op het zoekvak.

### .cta (\_blocks.scss)

Gebruik deze klasse op Call To Actions blokken.

### .flexslider (\_blocks.scss)

Deze klasse wordt standaard gebruikt op de slider op de homepagina.

### .contact (\_blocks.scss)

Deze klasse wordt standaard gebruikt op de wrapper van het contactformulier.

### .theaim (\_blocks.scss)

Deze klasse wordt standaard gebruikt op de logo van the AIM in de footer.

### .social (\_blocks.scss)

Deze klasse wordt standaard gebruikt op de wrapper van de social media iconen in de footer.

### p.intro (\_base\_include.scss)

Gebruik deze klasse op paragrafen van introteksten in de CKEditor. Deze klasse kan ook toegevoegd worden aan het bestand **ckeditor.styles.js**.

### p.clear (\_base\_include.scss)

Gebruik deze klasse op paragrafen die op een nieuwe lijn moeten beginnen. Deze klasse kan ook toegevoegd worden aan het bestand **ckeditor.styles.js**, zeer handig voor paragrafen die volgen op afbeeldingen.

### .colorbox

Gebruik deze klasse op wrappers van de link rond een afbeelding. Door deze klasse kan de afbeelding vergroot worden in een colorbox popup.

## Gebruik van klasses

### Generieke theming

Probeer theming altijd zo generiek (herbruikbaar) mogelijk te maken. Niemand schrijft graag meermaals dezelfde code, en dit zorgt er ook voor dat de code steeds overzichtelijk blijft. Plaats dus gerust meerdere klasses op een element, waarmee je de eerste klasse gebruikt voor algemene theming, die je ook voor andere elementen kan gebruiken, om met een andere klasse meer specifiek je ding te doen.

**Bijvoorbeeld**: In een website worden een aantal call to actions gebruikt. Uit de designs blijkt dat we 2 types hebben: een lichte en een donkere versie.



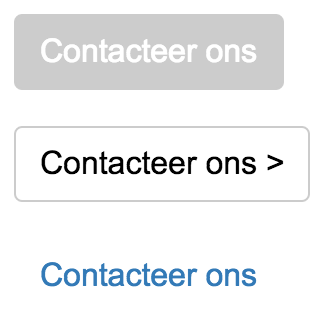
Opteer er dan voor om zoveel mogelijk algemene styling op 1 klasse te plaatsen, en met een andere klasse een paar eigenschappen te overschrijven voor de tweede versie.

.cta {  
 width: 100%;  
 float: left;  
 padding: rhythm(0.5);  
 margin-bottom: rhythm(1);  
 text-align: center;  
 background: $grey-light;  
 border: 1px solid $black;  
 border-radius: 10px;  
  
 p.title {  
 text-transform: uppercase;  
 }  
 a {  
 color: inherit;  
 text-decoration: underline;

&:hover {  
 text-decoration: none;  
 }  
 }  
}  
.cta--dark {  
 background: $grey-dark;  
 color: $white;  
}

Beide call to actions geven we de klasse .cta mee, en enkel de donkere versie geven we een extra klasse .cta--dark mee.

**Nog een voorbeeld:** In dezelfde website worden een aantal buttons gebruikt. Uit de designs blijkt dat we 3 types hebben.



Elke button krijgt de klasse .btn mee, waarop we de algemene styling gaan toepassen. Elk type button krijgt dan een eigen klasse mee, waarop we de meer specifieke styling doen.

/-------------------  
//---- BUTTONS ------  
//-------------------  
  
// General  
.btn {  
 display: inline-block;  
 border: 1px solid transparent;  
 border-radius: 4px;  
 color: $white;  
 padding: rhythm(0.25) rhythm(0.5);  
 text-decoration: none !important;  
}  
// Button default  
.btn--default {  
 color: $white;  
 background: $grey;  
 border-color: $grey;  
  
 &:hover {  
 background: $grey-light;  
 }  
}  
// Button more  
.btn--more {  
 color: $black;  
 background: $white;  
 border-color: $grey;  
  
 &:hover {  
 background: $grey-light;  
 }  
 &:after {  
 content: " \003E";  
 }  
}  
// Button link  
.btn--link {  
 color: $link;  
 background: transparent;  
  
 &:hover {  
 text-decoration: underline !important;  
 }  
}

### BEM klasses

In toekomstige versies van het install profile zullen we meer en meer gebruik gaan maken van BEM klasses. BEM, wat staat voor block, element, modifier - is een front-end naamgeving. Het is een slimme manier voor het benoemen van CSS klasses om hen meer transparantie en betekenis te geven voor andere ontwikkelaars. Ze zijn veel strenger en informatief, wat de BEM naamgeving ideaal maakt voor teams van ontwikkelaars.

De BEM naamgeving volgt onderstaand patroon:

.block {}  
.block\_\_element {}  
.block--modifier {}

* .block vertegenwoordigt het hogere niveau van een onderdeel.
* .block\_\_element is een afstammeling van .block.
* .block--modifier staat voor een andere staat of versie van .block.

De reden voor de dubbele in plaats van enkele koppeltekens en underscores is dat de block zelf koppeltekens en underscores kan bevatten. Bijvoorbeeld:

.search-box {} /\* Block \*/  
.search-box\_\_field {} /\* Element \*/  
.search-box--full {} /\* Modifier \*/

Het nut van BEM is om andere ontwikkelaars meer te vertellen over wat een stuk markup doet door zijn naam alleen. Door het lezen van stukken HTML met een aantal klassen in, kan je zien hoe de brokken gerelateerd zijn.

Het BEM principe is nog helemaal niet geïntegreerd in Install Profile 3.4, maar is zeer handig en aan te raden bij de uitwerking van een website. Probeer hier reeds zoveel mogelijk gebruik te maken.

Ook de voorbeelden in voorgaand puntje (Generieke theming), maken gebruik van dit principe.

# Verloop van een project

Hier worden de verschillende stappen die je moet doorlopen, bij de uitwerking van een website, kort beschreven.

## Lokale development opzetten

Haal de laatste versie van de bestanden via git binnen van de bitbucket server. Haal de database van het prototype binnen en koppel deze aan je lokale versie.

Maak vervolgens een dev branch aan, waarop je lokaal zal ontwikkelen en push regelmatig naar de bitbucket server. (<http://slides.com/davidcomeyne/gitworkflow#/3>)

Probeer regelmatig een commit toe te voegen, je commits te structureren en geef elke commit een duidelijke naam. (**Ok:** Basic page theming for mobile. **Niet ok**: Theming update)

## Bestudeer draaiboek en design

Haal het draaiboek van het project erbij, en neem dit kort even door. Zorg ervoor dat alles wat hierin beschreven staat duidelijk is vooraleer je start met de uitwerking van de website. Indien niet duidelijk, neem zeker contact op met je project coördinator of de persoon die het prototype heeft uitgewerkt.

Bestudeer ook grondig het design, zowel voor mobiel als desktop. Controleer allereerst even of je over alle designs beschikt, en of de nodige SVG afbeeldingen werden aangeleverd.

Denk bij het bestuderen van het design ook al na over opbouw, herbruikbare styling, BEM klasses, …

## Project variabelen

Start met de specifieke project variabelen in te vullen. Lettertypes, lettergrootte, kleuren, …

## Werk een eerste pagina uit.

Start met 1 pagina uit werken. Het handigste is om eerst een basic page (vb: over ons) uit te werken. Dit omdat zo een pagina heel wat algemene styling bevat, en de meeste onderdelen (header, footer, menu’s, …) hier aanwezig zijn en over de rest van de website terugkomen. De homepagina wijkt meestal af van de overige pagina’s en wordt daarom best niet als eerste uitgewerkt.

Denk hierbij aan het mobile first principe!

Wanneer de eerste pagina volledig klaar is (voor mobiel en desktop), plan dan een momentje in met de designer om deze pagina te overlopen. Zo kunnen onvolledigheden of foutjes al vroeg opgespoord en opgelost worden alvorens deze zich over het hele project verspreiden.

## Werk de overige pagina’s uit

Wanneer de designer zijn fiat gegeven heeft, kan je van start gaan met de uitwerking van de overige pagina’s van de website. Hou zeker de mindmap van het project erlangs, zodat je zeker bent dat je alle pagina’s behandelt hebt.

## Afwerking

Wanneer je klaar denkt te zijn met de uitwerking van de website, gaan we nog een aantal zaken controleren alvorens we de site op staging plaatsen:

* Basic tekst styles  
  Controleer of alle titels (h1 - h4), lijsten (ul, ol), paragrafen en introteksten correct weergegeven worden.
* 404 & 403  
  Controleer of er voor elke taal een 404 en 403 error pagina is ingesteld en correct wordt weergegeven. Controleer ook of de links correct liggen.
* Site administrator  
  Zorg ervoor dat er een gebruiker aangemaakt is voor de klant, met de juiste permissies. Test dit altijd uitgebreid.
* mag sommige content verwijderen,
* mag al zijn eigen content bewerken en verwijderen,
* mag sommige taxonomy terms aanmaken,
* mag formulierinzendingen bekijken
* Zorg op het dashboard voor een aantal dashboard links, zodat de klant vanuit 1 plaats alle acties die hij kan doen op de site kan uitvoeren.
* Inhoud bewerken
* Inhoud aanmaken (opsplitsen in verschillende content types)
* Formulierinzendingen bekijken
* Confimation mails  
  Controleer of voor elk formulier bevestigingsmails zijn ingesteld, zowel voor de site administrator als voor de invuller van het formulier, in alle beschikbare talen. Check voor shops ook de mails die verstuurd worden na doorlopen van het checkout proces.
* Error messages

Zorg ervoor dat alle error messages vertaald zijn naar de beschikbare talen.

* Bulk update content author

Zorg ervoor dat alle pagina’s, die door de klant aangepast kunnen worden, als auteur de site administrator hebben. Zo kan de klant, wanneer hij pagina’s op ‘niet gepubliceerd’ plaatst, ze toch nog bereiken en aanpassen.

Er is hiervoor binnen the AIM een module ontwikkeld, namelijk “Mass update owner”. Doe hiervoor het volgende:

* Schakel de module in.
* Selecteer de klantenlogin op /admin/config/the-aim/mass-update-content-owner
* Ga vervolgens naar het inhoudsoverzicht, selecteer de pagina’s waarvan je de eigenaar wilt wijzigen, en selecteer in de update opties de optie “Change author to siteowner”
* Imagefield setups  
  Controleer of elk voor afbeeldingsveld een minimum en maximum formaat, en een ‘file directory’ is ingesteld.
* Search

Controleer of de zoekfunctie (indien aanwezig) correct werkt.

* Site informatie checken  
  Vul het e-mail adres van de klant in op /admin/config/system/site-information.
* Privacy policy  
  Vul voor elke beschikbare taal de privacy policy aan met de gegevens van de klant.
* URL aliasses  
  Controleer of de URL aliasses correct zijn ingesteld voor elk content type.
* Breadcrumbs  
  Controleer of de breadcrumbs kloppen voor elke pagina en of ze betekenisvol zijn.
* XML sitemap  
  Controleer of de nodige pagina’s correct verschijnen in de XML sitemap. De 404 & 403 pagina’s dienen hier niet in voor te komen.
* Favicon & Touch icons  
  Zorg ervoor dat het favicon en de touch iconen zijn ingesteld.
* Printmarkup

Controleer hoe de verschillende pagina’s eruit zien wanneer je ze wil afprinten. Zorg ervoor dat voor elk type pagina enkel de belangrijke content zal worden afgeprint. Onderdelen zoals logo, menu’s, call to actions, … zijn niet altijd nodig bij het afprinten. Voeg, indien nodig, styling toe aan het bestand css/print.php om bepaalde onderdelen te verbergen.

* Colorbox styling  
  Controleer of de colorbox gestyled is in de huisstijl van de website.
* Animations, hovers, focus state  
  Zorg voor animaties, hovers en focus states op links.
* Devel & coffee module  
  Controleer de logberichten in de backoffice. Als deze in orde zijn, schakel de Devel & Coffee module uit.
* the AIM environmental variable aanpassen

Plaats the AIM environmental variable op ‘Staging’. Deze setting kan aangepast worden op “the AIM general settings page” (/admin/config/the-aim/settings).

* Caching inschakelen
* Pagina’s cachen voor anonieme gebruikers
* Blokken cachen
* Minimum cache levensduur: 3 uur
* Verval van gecachte pagina’s: 15min
* Gecachte pagina’s comprimeren
* JavaScript bestanden samenvoegen
* Crossbrowser check

Controleer de website in Google Chrome, Mozilla Firefox, Safari, IE8, IE9, IE10, EI11 en Opera.

Wanneer bovenstaande puntjes allemaal in orde zijn, kan de site op staging geplaatst worden.

Maak ook een ticketje aan in het board ‘Webmaster Planning’ in trello, op basis van het ticket ‘Checklist’ onder de kolom ‘sjablonen’. Vul de naam van de klant hierop in, zet in de beschrijving van het ticket de staging url, en koppel je projectmanager en een andere webmaster aan dit ticket. Deze webmaster zal de eerste technische check uitvoeren.

Wanneer er nog zaken aangepast dienen te worden aan de website, zal je via dit ticket op de hoogte gebracht worden.