|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 11-11-2016 |  | | |  | |
|  | |  | | |  | |
|  | | | Sass vs LESS  *VERGELIJKENDE STUDIE* | | | |
|  |  | | |  | |  |
|  |  | | |  | | Lander Verschueren |

Inhoud

[Introductie 2](#_Toc466571113)

[Installatie 3](#_Toc466571114)

[Code 3](#_Toc466571115)

[Nesting 3](#_Toc466571116)

[Mixins 4](#_Toc466571117)

[Extend 7](#_Toc466571118)

[Operations 8](#_Toc466571119)

[Extensions 9](#_Toc466571120)

[Loops 9](#_Toc466571121)

[Media Queries 11](#_Toc466571122)

[Conclusie 12](#_Toc466571123)

[Bronnen 13](#_Toc466571124)

Sass vs LESS

VERGELIJKENDE STUDIE

# Introductie

Sass en LESS, beide vallen onder de categorie CSS preprocessors. Preprocessors zijn programma’s die een bepaald bestandstype omzet naar een ander type. In dit geval van een .scss- of .less-bestand naar een bruikbaar .css-bestand dat door alle moderne browsers ondersteund wordt. De populairste van deze twee is zonder twijfel Sass, dat wordt gebruikt door 66% van de ondervraagde developpers tegenover 13,4%. In deze vergelijkende studie zet ik de verschillen in installatie, code en een aantal andere dingen naast elkaar. Maar eerst een klein woordje over preprocessors.

*“In computer science, a preprocessor is a program that processes its input data to produce output that is used as input to another program. The output is said to be a preprocessed form of the input data, which is often used by some subsequent programs like compilers. The amount and kind of processing done depends on the nature of the preprocessor; some preprocessors are only capable of performing relatively simple textual substitutions and macro expansions, while others have the power of full-fledged programming languages.”*<https://en.wikipedia.org/wiki/Preprocessor>

Voor CSS preprocessor komt dit naar op een aantal voordelen:

* Minder lijnen code.
* Overzichtelijkere code.
* Hergebruik van code.
* …

# Installatie

Sass en LESS zijn op een verschillend platform gebouwd. Waar Sass op Ruby draait is LESS gemaakt in JavaScript.

Voor Sass wordt er dus van de gebruiker verwacht dat hij Ruby geïnstalleerd heeft op zijn computer. Voor Mac gebruikers is dit geen probleem aangezien Ruby aanwezig is. Windows gebruikers zullen eerst Ruby nog moeten installeren, wat een pain-in-the-ass kan zijn. Verder installeer je Sass altijd via Ruby.  
Al is Sass ook al overgebracht naar een versie op LibSass (C++), wat het gebruik en de prestaties alleen maar ten goede komt.

LESS daartegenover is zoals eerder vermeld op JavaScript gebouwd en is hierdoor veel simpeler in installatie. Het enige dat je nodig hebt is Node.js.

Verder kan je voor beide applicaties gebruiken die het gebruik van vorige methodes onnodig maakt. Enkele voorbeelden hiervan zijn CodeKit, Koala en LiveReload.

Het is ook aangewezen om een plugin te installeren in je editor om de juiste highlights te geven aan je code zodat het minder vlak lijkt.  
Voor SublimeText: Sass Bundle of Less-sublime, Notepad++: Notepad-plus-plus of Less-for-Notepad-plusplus, Coda: Coda Sass of Coda 2 Less.

# Code

Beide preprocessors hebben hun eigen taal, net zoals jQuery gebruikt Sass een dollarteken ($) om hun variabelen te definiëren terwijl LESS een hashtag (#) gebruikt om hetzelfde te doen. Wat volgt zijn een aantal principes die zowel binnen Sass als LESS gebruikt worden.

## Nesting

Je kan nesting het best vergelijken met HTML-code waarin elk element binnen een ander element geplaatst wordt. Zo kan je bij zowel bij beide preprocessors selectors binnen een hogere selector plaatsen net zoals bij HTML-code. Bijde prepocessors compilen dit op eenzelfde manier.

#main p {

color: #00ff00;

width: 97%;

.redbox {

background-color: #ff0000;

color: #000000;

}  
}

Gecompileerd:

#main p {

color: #00ff00;

width: 97%; }

#main p .redbox {

background-color: #ff0000;

color: #000000; }

Sass biedt je wel een iets diepere mogelijkheid aan, de mogelijkheid om binnen selectors individuele properties te zetten.

.funky {

font: {

family: fantasy;

size: 30em;

weight: bold;

}

}

Gecompileerd:

.funky {

font-family: fantasy;

font-size: 30em;

font-weight: bold; }

## Mixins

Hier zit waarschijnlijk een van de grootste verschillen tussen Sass en LESS. Sass gebruikt @mixin terwijl LESS gebruikt maakt van class selectors.

Sass:

@mixin large-text {

font: {

family: Arial;

size: 20px;

weight: bold;

}

color: #ff0000;

}

.page-title {

@include large-text;

padding: 4px;

margin-top: 10px;

}

Gecompileerd:

.page-title {

font-family: Arial;

font-size: 20px;

font-weight: bold;

color: #ff0000;

padding: 4px;

margin-top: 10px; }

LESS:

.a, #b {

color: red;

}

.mixin-class {

.a();

}

.mixin-id {

#b();

}

Gecompileerd:

.a, #b {

color: red;

}

.mixin-class {

color: red;

}

.mixin-id {

color: red;

}

Waar je wel rekening mee moet houden bij LESS is dat deze selectors geïnterpreteerd worden als css-selectors eens deze gecompileerd zijn. Als je de mixins niet als output wil moet je haakjes na je mixin zetten.

.my-mixin {

color: black;

}

.my-other-mixin() {

background: white;

}

.class {

.my-mixin;

.my-other-mixin;

}

Gecompileerd:

.my-mixin {

color: black;

}

.class {

color: black;

background: white;

}

## Extend

Mogelijk in zowel Sass als LESS maar werkt op een iets andere manier. Daar waar bij LESS de stijlen van de ene selector als het ware worden geïmporteerd in de andere selector, wordt bij Sass de selector dat ge-extend wordt aangepast met een tweede selector.

Sass:

.message {

border: 1px solid #ccc;

padding: 10px;

color: #333;

}

.success {

@extend .message;

border-color: green;

}

.error {

@extend .message;

border-color: red;

}

.warning {

@extend .message;

border-color: yellow;

}

Gecompileerd:

.message, .success, .error, .warning {

border: 1px solid #cccccc;

padding: 10px;

color: #333;

}

.success {

border-color: green;

}

.error {

border-color: red;

}

.warning {

border-color: yellow;

}

LESS:

nav ul {

&:extend(.inline);

background: blue;

}

.inline {

color: red;

}

Gecompileerd:

nav ul {

background: blue;

}

.inline,

nav ul {

color: red;

}

## Operations

Beide preprocessors kunnen wiskundige berekeningen uitvoer en doen dit op dezelfde manier.

Het enige verschil tussen beide is dat beide berekeningen tussen 2 verschillende waarden (bv.: pixels en procenten) Sass een syntax error gaat geven tegenover LESS dat gewoon 0px gaat teruggeven.

## Extensions

Sass heeft hier de bovenhand en dat heeft het enkel en alleen te danken aan Compass. Compass is een framework bovenop Sass dat allerhande extensies bundelt. Bij LESS moet je deze allemaal zelf importeren.

## Loops

Sass heeft hier een grote voorsprong of LESS que mogelijkheden. Waar LESS stopt bij simpele loops als:

.loop(@counter) when (@counter > 0) {

.loop((@counter - 1)); // next iteration

width: (10px \* @counter); // code for each iteration

}

div {

.loop(5); // launch the loop

}

.generate-columns(4);

.generate-columns(@n, @i: 1) when (@i =< @n) {

.column-@{i} {

width: (@i \* 100% / @n);

}

.generate-columns(@n, (@i + 1));

}

Gaat Sass een stuk verder met volledige support voor if-, for- en each-statements.

p {

@if 1 + 1 == 2 { border: 1px solid; }

@if 5 < 3 { border: 2px dotted; }

@if null { border: 3px double; }

}

Gecompileerd:

p {

border: 1px solid; }

$type: monster;

p {

@if $type == ocean {

color: blue;

} @else if $type == matador {

color: red;

} @else if $type == monster {

color: green;

} @else {

color: black;

}

}

Gecompileerd:

p {

color: green; }

@for $i from 1 through 3 {

.item-#{$i} { width: 2em \* $i; }

}

Gecompileerd:

.item-1 {

width: 2em; }

.item-2 {

width: 4em; }

.item-3 {

width: 6em; }

@each $animal in puma, sea-slug, egret, salamander {

.#{$animal}-icon {

background-image: url('/images/#{$animal}.png');

}

}

Gecompileerd:

.puma-icon {

background-image: url('/images/puma.png'); }

.sea-slug-icon {

background-image: url('/images/sea-slug.png'); }

.egret-icon {

background-image: url('/images/egret.png'); }

.salamander-icon {

background-image: url('/images/salamander.png'); }

## Media Queries

Vroeger waren deze omslachtig en stonden deze apart in je css, wat niet zo overzichtelijk was. Met Sass en LESS kan je deze media queries gewoon in je selectors zetten.

.some-class {

/\* Default styling \*/

@media (max-width: 800px) {

/\* Responsive styles \*/

}

}

# Conclusie

Eigenlijk zijn deze twee preprocessors aan elkaar gewaagd met hier en daar kleine puntjes van verschil. Het enige verschil is dat Sass veel wijdverspreider is dan LESS en dus meer dan waarschijnlijk ook een grotere support heeft.

Alsook heeft Sass het voordeel dat een groot deel van extensies niet apart ingeladen moet worden door Compass. Alsook biedt Sass een iets bredere functionaliteit ten opzichte van LESS, wat misschien voor sommige mensen overweldigend kan zijn en de learning curve niet ten goede komt.

Mijn voorkeur zou naar Sass uitgaan. Hierin trekt de functionaliteit van loops me over de streep, maar dat is ook zowat het enige.

De keuze tussen Sass en LESS blijft dus veelal een persoonlijke voorkeur!

# Bronnen

[https://www.keycdn.com/blog/Sass-vs-less/](https://www.keycdn.com/blog/sass-vs-less/)

[http://www.hongkiat.com/blog/Sass-vs-less/](http://www.hongkiat.com/blog/sass-vs-less/)

<https://css-tricks.com/sass-vs-less/>

<https://en.wikipedia.org/wiki/Preprocessor>

<http://lesscss.org/features/>

[http://Sass-lang.com/documentation/](http://sass-lang.com/documentation/)