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

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. 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.

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.

# 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;

}  
}

#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;

}

}

.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;

}

.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();

}

.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;

}

.my-mixin {

color: black;

}

.class {

color: black;

background: white;

}