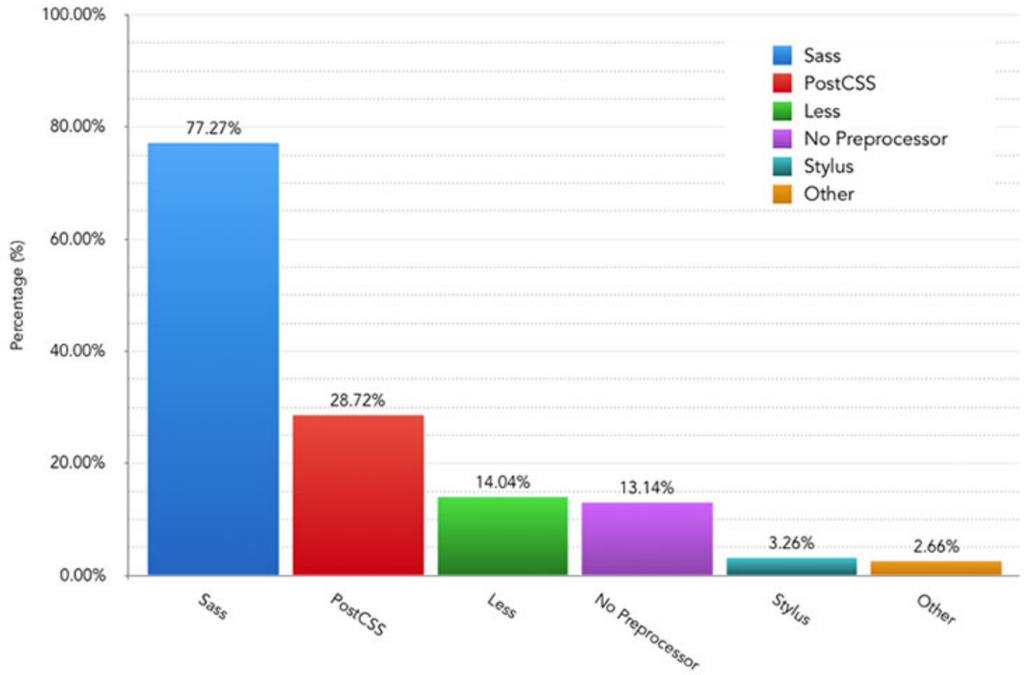
Syntactically Awesome StyleSheets

Theoretisch

Sass = populairste preprocessor



Preprocessor statistieken herfst 2019

https://2020.stateofcss.com/en-US/technologies/pre-post-processors/

Sass = CSS op steroïden

- abstractielaag, meer logica
- voegt mogelijkheden van programmeertalen toe aan CSS
- extra functionaliteit, geavanceerde CSS
- DRY - minder code schrijven, efficiënter
- beter te beheren en te onderhouden
- makkelijker uit te breiden
- …

Sass = super power!

- variables
- nesting
- mixins - @include directive
- extensions - @extend directive
- functions
- operators
- lists & maps (=arrays)
- interpolation
- for, each and while-loops & if-else statements
- partials / modules
- ~~@import~~ - nieuw: @use en @forward
- …

Syntax - 2 smaken

.sass (indented)

Originele syntax

```
$font-stack: Helvetica, sans-serif
$primary-color: #333

body
  font: 100% $font-stack
  color: $primary-color
```

.scss (“Sassy” 😎😛 CSS)

Superset van CSS

```
$font-stack: Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

Van Sass naar CSS

- browsers kunnen Sass niet interpreteren
- daarom moeten we Sass compileren (“transpileren”) naar CSS
- geldige CSS = geldige SCSS; je kan dus “vanilla” CSS schrijven in SCSS bestanden
- we linken naar het resulterende CSS bestand in de head, zoals gewoonlijk, niet naar onze SCSS bestanden
- we passen nooit code aan in de CSS output, we werken enkel in onze SCSS stylesheets!!

Compilers

- CLI
- task runners: Gulp, Grunt
- online compilers: SassMeister,…
- apps: Prepros, Koala, CodeKit,…
- IDE (packages, extensions)

GRUNT

New: Sass Module System

- Bestaat sinds oktober 2019
- Maakt gebruik van **@use & @forward** voor imports van partials en built-in Sass modules i.p.v. @import
- @import wordt uitgefaseerd in oktober 2021 en support verloopt in oktober 2022
- Werkt enkel met **Dart Sass**

Module System compiler support

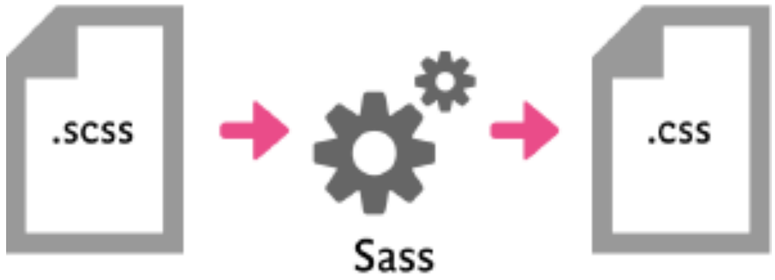
- **Dart Sass** is momenteel de primaire implementatie van Sass
- De Live Sass Compiler in VS Code gebruikt LibSass. Maar LibSass biedt geen module system support dus deze optie kunnen we niet gebruiken… 😢
- In oktober 2020 werd aangekondigd dat ook LibSass en Node Sass uitgefaseerd gaan worden
- Het veel gebruikte Ruby Sass kende z'n end of life in april 2019
- We zitten momenteel (februari 2021) dus in een **overgangsfase**
- We gaan ons nu al helemaal focussen op de latest & greatest!

Dart Sass

- Online compiler: **SassMeister**
- App: **Prepros**
- Command line: **Terminal** op Mac, iTerm, **Windows cmd**, PowerShell, Git Bash,…
- Taskrunner: **Gulp** (in de lessen van Bert)
- IDE:
  - VS Code: wachten op een Dart Sass compiler extension… (console kan wel gebruikt worden)
  - Sublime Text 3: er bestaat een workaround, zie https://stackoverflow.com/questions/53642701/sublime-text-3-sass-build-error-cannot-build-a-css-file/56479702#56479702

Praktisch

Online oefenen: SassMeister



Output styles

- nested (default)
- expanded
- compact
- compressed

Variables

- beginnen met een $
- kunnen iedere CSS waarde bevatten
- waardes die meermaals voorkomen
- snel centraal aanpasbaar
- Let op, Sass variables compileren niet naar native CSS variables maar naar een waarde

Variables

```
$title-font: normal 24px/1.5 'Open Sans', sans-serif;
$cool-red: #f44336;
$box-shadow-bottom-only: 0 2px 1px 0 rgba(0, 0, 0, 0.2);

h1.title {
  font: $title-font;
  color: $cool-red;
}

div.container {
  color: $cool-red;
  background: #fff;
  width: 100%;
  box-shadow: $box-shadow-bottom-only;
}
```

Nesting

- beknoptere code
- maximum 4 niveaus diep, hou het best bij 3
- let op met de accolades!

Nesting

```
ul {
  list-style: none;

  li {
    padding: 15px;
    display: inline-block;

    a {
      text-decoration: none;
      font-size: 16px;
      color: #444;
    }
  }
}
```

Nesting

```
.your_new_class {
  background: orange;
  p {
    font: {
      weight: bold;
      size: 14px;
      family: times;
    }
  }
}
```

=> nested properties - overeenkomstige namespaces

Nesting

```
a {
  color: red;
  text-decoration: none;
  &:hover {
    text-decoration: underline;
  }
}
```

=> pseudo-classes

Nesting

```
button {
  background: gray;
  border: 1px solid black;
  border-radius: 5px;
  &.fail {
    color: red;
    border-color: red;
  }
}
```

=> parent selector

Nesting

```
.box {
  box-sizing: border-box;
  width: 800px;
  padding: 30px;
  border: 1px solid black;
  .no-boxsizing & {
    width: 738px;
  }
}
```

Mixins

```
@mixin center() {
  display: flex;
  justify-content: center;
  align-items: center;
}

.centered-elements {
  @include center();
  background-color: #f9f9f9;
}
```

Mixins with an argument

```
@mixin center($direction) {
    display: flex;
    justify-content: center;
    align-items: center;
    flex-direction: $direction;
}

div {
    @include center(column);
}
```

Mixins with multiple arguments

```
@mixin border($radius, $style) {
  border-radius: $radius;
  border: $style;
}


.my-new-class {
  @include border(3px 3px 0 0, 1px solid orange);
}


.my-other-class {
  @include border(5px, 2px dotted red);
}
```

Mixins with default values

```
@mixin border($radius: 3px, $style: 1px solid red) {
  border-radius: $radius;
  border: $style;
}


.my-new-class {
  @include border;
}

.my-other-class {
  @include border(3px 3px 0 0, 1px solid orange);
}
```

Mixins with variable arguments

```
@mixin box-shadow($shadows...) {
  box-shadow: $shadows;
}

.shadows {
  @include box-shadow(0px 4px 5px #666, 2px 6px 10px #999);
}
```

Selector inheritance/extending rulesets

```
.dialog-button {
  box-sizing: border-box;
  color: #fff;
  box-shadow: 0 1px 1px 0 rgba(0, 0, 0, 0.12);
  padding: 12px 40px;
  cursor: pointer;
}

.confirm {
  @extend .dialog-button;
  background-color: #87bae1;
  float: left;
}

.cancel {
  @extend .dialog-button;
  background-color: #e4749e;
  float: right;
}
```

Selector inheritance with placeholder

```
%card {
    box-sizing: border-box;
    border-top: 1px rgba(#000, .12) solid;
    width: 100%;

    &:hover { border: 2px rgba(#000, .5) solid; }
}

.card-light {
    // inheriting card styling rules
    @extend %card;
    background-color: #fff;
    color: #000;
}

.card-dark {
    // inheriting card styling rules
    @extend %card;
    background-color: #000;
    color: #fff;
}
```

Selector inheritance with placeholder

```
%large-type {
  font-size: 3.83333em;
  line-height: 1.17391em;
  margin-bottom: 0.3913em;
  color: #333333;
}


h1 {
  @extend %large-type;
}


h2 {
  @extend %large-type;
  font-size: 2.66667em;
}
```

=> % = placeholder, wordt nooit een selector

Operators

```
+ addition
- subtraction
* multiplication
/ division
% modulo
```

- zonder calc()
- oppassen met min() - conflicten tussen Sass- en CSS functies

Operators

```
$width: 800px;
.container {
  width: $width;
}
.column-half {
  width: $width / 2;
}
.column-fifth {
  width: $width / 5;
}

$text-size: 15px;
#normal-text {
    font-size: $text-size;
}
#large-text {
    font-size: $text-size * 1.5;
}
```

=> er zijn nog andere operatoren dan rekenkundige (toewijzings-, vergelijkings- en logische operatoren)

Built-In Sass functions

```
$purple: #6A67CE;
$darkpurple: darken($purple, 15%);

.purple-btn {
    text-align: center;
    background-color: $purple;
}

.purple-btn:hover {
    background-color: $darkpurple;
}
```

=> raadpleeg de officiële Sass site voor een overzicht van alle ingebakken functies; er zijn 7 built-in modules: https://sass-lang.com/documentation/modules

Custom functions

```
@function my-calculation($some-number, $another-number){
  @return $some-number + $another-number;
}

.my-module {
  padding: my-calculation(10px, 5px);
}

@function text-color($bg-color) {
    @if (lightness($bg-color) > 60) { // lightness is een built-in Sass function
        @return #555;
    }
    @else {
        @return #fff;
    }
}
```

=> output is een enkele waarde - bij mixins kan de output meerdere CSS regels beslaan

Mixins, extensions of functions?

- **Mixins:** een aantal gelijkaardige eigenschappen die we meerdere keren gebruiken met kleine variaties
- **Extensions:** een aantal eigenschappen die exact matchen
- **Functions:** operations die een waarde bepalen

Comments in Sass

- `// single line comment wordt niet gecompileerd`
- `/* deze comment wordt mee gecompileerd */`
- `/*! dit is een comment die zelfs compressed mee gecompileerd wordt */`

Partials/modules

- i.p.v. al onze code in één lang SCSS bestand te schrijven, delen we onze code op in verschillende SCSS stylesheets
- deze vormen dan logische componenten, die we partials/modules noemen
- heel praktisch bij grote projecten en/of projecten waar je met een team aan werkt
- partials/modules beginnen altijd met een underscore
- bvb: _reset.scss, _variables.scss, _menu.scss, _responsive.scss,…
- partials/modules worden niet apart gecompileerd in een CSS bestand

Partials/modules importeren

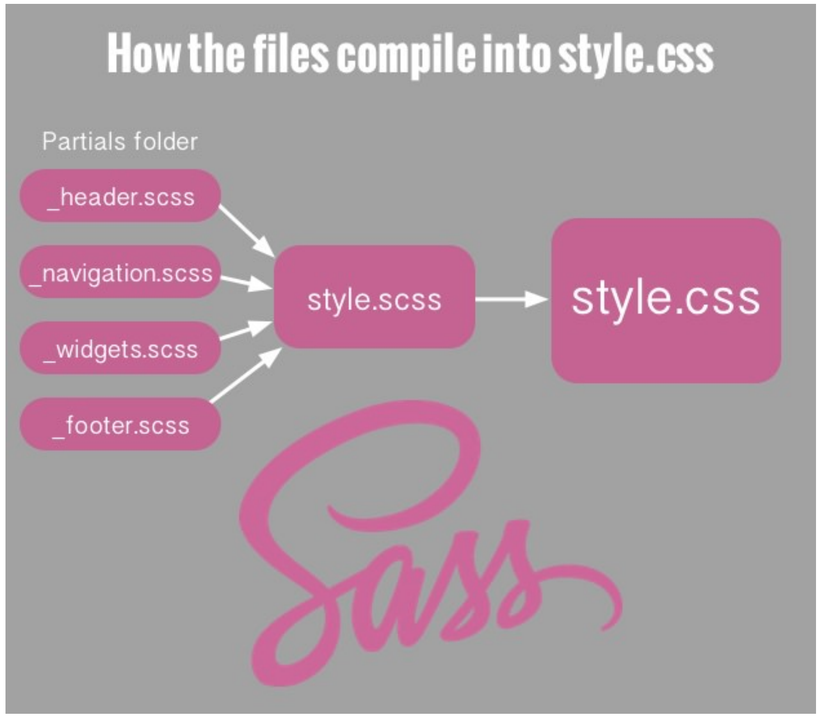
- partials/modules importeer je bovenaan in een main.scss bestand met @use
- bvb: @use ‘variables’; (als eerste regel in main.scss - underscore en extensie zijn optioneel bij het importeren)
- wil je code uit een partial/module (bvb. variables) in een andere partial/module gebruiken, moet je die ook daarin importeren met @use
- de variables, mixins, functions,… uit een partial/module noemen we members
- met de @forward at-rule kan je de members uit modules die geïmporteerd zijn met de @use at-rule beschikbaar maken wanneer de stylesheet in een andere stylesheet is opgenomen

@use & @forward

- meer hierover in een volgende les
- deze presentatie wordt nog verder aangevuld 😇

Partials/modules importeren

- Import gebeurt voor compiling dus goed voor performance =>minder http requests



The 7–1 architecture

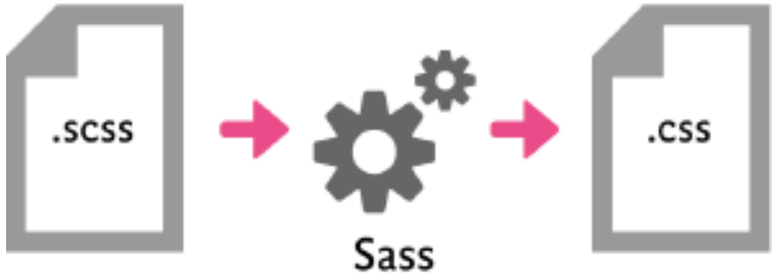
- een populaire architectuur om je Sass bestanden te organiseren
- logische structuur die erg geschikt is voor grote projecten
- we maken max. 7 folders aan en één main Sass bestand in de root waarin we alle folders met partials importeren
- bestudeer het voorbeeld, you'll get the idea 🤓

```
Sass/
|
|– abstracts/
|   |– _variables.scss    # Sass Variables
|   |– _mixins.scss       # Sass Mixins
|   |– _functions.scss    # Sass Functions
|
|– vendors/
|   |– _icons.scss        # Font-Awesome Icons
|
|– base/
|   |– _reset.scss        # Reset
|   |– _typography.scss   # Typography rules
|   |– _animations.scss   # Animation rules
|   |– _utilities.scss    # Utility rules
|
|– layout/
|   |– _navigation.scss   # Navigation
|   |– _header.scss       # Header
|   |– _footer.scss       # Footer
|
|– components/
|   |– _buttons.scss      # Buttons
|   |– _card.scss         # Card
|   |– _form.scss         # Form
|
|– pages/
|   |– _home.scss         # Home page style
|   |– _contact.scss      # Contact page style
|   |– _about.scss        # About page style
|
|– themes/
|   |– _pink.scss         # Pink theme
|   |– _mint.scss         # Mint theme
|
 – main.scss              # Main Sass input file
```

Sublime Text & VS Code

- Installeer minstens de Sass extension/package
- Een IDE compiler package/extension voor Dart Sass is er momenteel dus (nog) niet…

Oefenen met de Prepros app



Compileren met Prepos

- download Prepros: https://prepros.io
- we overlopen de settings
- we compileren een bestaand project: download de Heartbeat opdracht

Source map

- .map bestand in JSON formaat
- linkt iedere CSS regel (output) aan de corresponderende regel in het SCSS bronbestand (input) => handig bij inspectie in je browser
- onderaan je CSS bestand vind je deze regel terug na compilatie:

```
/*# sourceMappingURL=style.css.map */
```

Refactoring

- “vanilla” CSS omzetten naar Sass
- neem een bestaande project folder, maak daarin een nieuwe folder aan en noem deze “scss”
- maak daarin een nieuw bestand aan en noem het “main.scss”
- kopieer en plak de inhoud van je bestaande CSS stylesheet in “main.scss”
- verwijder je originele css folder en inhoud
- herschrijf je CSS naar Sass

Een nieuw Sass project starten

- maak een project folder aan
- maak daarin een nieuwe folder aan en noem deze “scss”
- maak daarin een nieuw bestand aan en noem het “main.scss”
- maak een index.html bestand aan en link alvast naar “main.css”
- de compiler maakt een “css” folder aan met daarin een “main.css”

Let’s get Sassy!

- maak een project folder aan en noem deze “je-naam-gets-sassy”
- maak daarin een nieuwe folder aan en noem deze “scss”
- maak daarin een nieuw bestand aan en noem het “main.scss”
- download de “Let’s get Sassy” opdracht en plaats het HTML bestand in de root van je project folder
- we schrijven alle stijlregels in eerste instantie in het “main.scss” bestand
- daarna passen we de 7-1 architecture toe

Oefenen met de command line

- Open Terminal op Mac (of de command prompt op Windows)
- Check of je npm al geïnstalleerd hebt, en welke versie, met het commando npm -v
- Nog niet geïnstalleerd? Download en installeer dan eerst Node.js: https://nodejs.org/en/
- Check vervolgens of Sass al geïnstalleerd is, en welke versie, met het commando sass —version
- Nog niet geïnstalleerd? Doe dat dan nu met het commando npm install -g sass of sudo npm install sass (Mac/Linux)
- Maak een nieuwe project folder aan, met daarin een scss folder met daarin een leeg main.scss (of style.scss) bestand, en sleep je project folder alvast in je editor
- Dan zorg je dat de Terminal/Windows-prompt in je project folder komt te staan. Je geeft het commando cd met een spatie en sleept je project folder op de CLI en entert dan.
- Typ dan het commando sass --watch scss:css
- De CLI genereert automatisch een css folder met daarin een main.css (of style.css) bestand, slaat alle wijzigingen die je maakt op en compileert die naar je css bestand, waarnaar je koppelt in de head van je HTML bestand, zoals gewoonlijk
- Stoppen met watchen kan met Ctrl + C

@debug

- meer hierover in een volgende les
- deze presentatie wordt nog verder aangevuld 😇