Sass workshop

1 Wat is SASS?

SASS (Syntactically Awesome Style Sheets) is een CSS extension language. Het voegt extra features toe aan CSS. Tot voor kort ondersteunde CSS geen variabelen, maar in SASS bestaan variabelen al meer dan 10 jaar. En ook nu we 'native' in CSS kunnen werken met variabelen blijft het gebruik van SASS zinvol, zeker voor grotere projecten. Bovendien biedt SASS naast variabelen ook nog veel andere features zoals interpolation, nested rules, mixins, partials, flow control, … Zo zal je jouw CSS, in grotere projecten, niet in één file schrijven, maar zal je meer modulair werken en je CSS opsplitsen over meerdere files. In zo een geval is het zeker handig om SASS te gebruiken.

Deze workshop is slechts een inleiding in het werken met SASS en we zullen enkel de volgende mogelijkheden van SASS bespreken:

- Variabelen
- Nested rules
- Partials

SASS ondersteunt twee verschillende syntaxen. De SCSS-syntax gebruikt de bestandsextensie .scss en de SASS-syntax gebruikt de bestandsextensie .sass.

De SCSS syntax is een superset van CSS, wat betekent dat elke geldige .css-file ook een geldige .scss-file is. Gezien zijn gelijkenis met CSS is de SCSS-syntax het populairst en we zullen in deze workshop dan ook de SCSS-syntax gebruiken.

2 Variabelen

De namen van variabelen moeten in SASS beginnen met een $-teken en om een expressie toe te kennen aan een variabele gebruiken we in SASS de dubbele punt.

Een variable declaration, voorbeeld:

```
$primary-color: rgb(173, 178, 247);
```

lijkt dus op een CSS declaration.

Je kan verschillende soorten informatie (value types) opslaan in een SASS-variabele. Onder andere getallen, strings, kleuren en lijsten met waarden.

Extra info: welke 'value types' SASS allemaal ondersteunt vind je op de officiële SASS-website https://sass-lang.com/documentation/values.

1. Open de map **sass-workshop** in Visual Studio Code
2. Creëer in de map **styles** een bestand **styles.scss**. Merk op dat Visual Studio Code een ander file-icon gebruikt voor een .scss-file () dan voor een .css-file ( ).
3. Voeg de volgende SASS-code toe aan **styles.scss**.

```scss
// Variabelen
// **********

$primary-color: rgb(173, 178, 247);
$light: white;
$font: Calibri, sans-serif;

// Basis stijlen
// *************

body {
  font-family: $font;
}

h1 {
  text-align: center;
}

body, header {
  background-color: $primary-color;
}

.container {
  background-color: $light;
}

header, .container {
  padding: 10px;
}

.container {
  max-width: 600px;
  margin: auto;
}
```

Zoals je hierboven al kan zien zullen de meeste SASS stylesheets, net als CSS stylesheets, voornamelijk bestaan uit style rules die css declarations bevatten, maar SASS stylesheets kunnen daarnaast ook variabelen-declaraties , mixins, enz. bevatten.

Daar browsers geen SCSS-code ondersteunen zullen we, als we het resultaat van bovenstaande SCSS-code willen zien, deze eerst moeten omzetten naar CSS. Dit gebeurt via een preprocessor in ons geval de SASS-compiler . De eenvoudigste manier om dit te doen is gebruikmaken van de Visual Studio Code extensie **Live Sass Compiler**.

Als de **Live Sass Compiler**-extensie correct geïnstalleerd is in Visual Studio Code dan zal je onderaan op de statusbalk **Watch Sass** zien staan.

Als dit niet het geval is, moet je de extensie eerst installeren (zie eerste les Benodige software).

4. Klik op **Watch Sass**. Hierdoor zal je **styles.scss**-bestand omgezet worden naar een **styles.css**-bestand en bovendien zal er ook nog een extra bestand **styles.css.map** aangemaakt worden.

   Je kan dit aflezen In het OUTPUT-venster:

   ```
   Generated:
   ... \sass-workshop\styles\styles.css.map
   ... \sass-workshop\styles\styles.css
   --------------------
   Watching...
   --------------------
   ```

   Als dit niet het geval is, heb je waarschijnlijk een typfout gemaakt en zal je in het OUTPUT-venster wellicht meer info over de 'error' kunnen aflezen.

   Vanaf nu zal de Live Sass compiler telkens als je **styles.scss** bewaart de twee files **styles.css** en **styles.css.map** opnieuw genereren.

   Extra info: het bestand **styles.css.map** is een source map-file en dient om ervoor te zorgen dat je in de Browser Developer Tools de SCSS-code kan bekijken. Als je **styles.css** opent in Visual Studio Code zal je zien dat op het einde van de file er een verwijzing is naar de bijbehorende source map-file `/*#sourceMappingURL=styles.css.map */`

5. Bekijk de door de SASS-compiler gegenereerde CSS-code in het gecompileerde bestand **styles.css**.
   Merk o.a. op dat de commentaarregels niet overgenomen zijn in **styles.css**.
   SASS ondersteunt twee types commentaar. Commentaar aangeduid met de scripting syntax `//` is niet zichtbaar in de gecompileerde file. Commentaar aangeduid met de CSS-syntax `/* */` zal wel zichtbaar zijn in de gecompileerde file.

6. Leg vervolgens een link in je html-bestand naar het bestand **styles.css** en controleer of je het onderstaande resultaat bekomt.

Extra Info: voor meer info over SASS-variabelen verwijzen we naar de officiële SASS-website: https://sass-lang.com/documentation/variables.

3 Nested rules

3.1 Nested rules ter vervanging van CSS combinators

In SASS kan je stijlregels nesten. Zo kan je bijvoorbeeld de volgende CSS-code

```
nav ul {
    ...
}
nav a {
    ...
}
```

met behulp van 'nested rules' in SASS ook als volgt schrijven:

```
nav {
    ul {
        ...
    }

    a {
        ...
    }
}
```

7. Voeg aan **styles.scss** de SASS-code uit de linkerhelft van onderstaande tabel toe. Controleer vervolgens in **styles.css** dat deze code omgezet wordt naar de CSS-code uit de rechterhelft van de tabel. Negeer hierbij de CSS properties met '[vendor prefixes]'. Deze laatsten hebben niets te zien met de SASS-compiler, maar de Visual Studio Code extensie **Live Sass Compiler** bevat naast een SASS-compiler ook een autoprefixer (voor meer info zie: [CSS3 Compatibility & Vendor Prefixes]) en deze regels zijn daarvan afkomstig.

SCSS Source code	Compiled CSS code

`// Navigatiebalk`
`// *************`
 	
`nav {`
`  ul {`
`  margin: 0;`
`  padding: 0;`
`  list-style-type: none;`
`  /* navigatiemenu horizontaal maken */`
`  display: flex;`
`  justify-content: space-between;`
`  }`
 	
`  a {`
`  display: block;`
`  color: $light;`
`  }`
`}`
	`nav ul {`
	`  margin: 0;`
	`  padding: 0;`
	`  list-style-type: none;`
	`  /* navigatiemenu horizontaal maken */`
	`  display: -webkit-box;`
	`  display: -ms-flexbox;`
	`  display: flex;`
	`  -webkit-box-pack: justify;`
	`  -ms-flex-pack: justify;`
	`  justify-content: space-between;`
	`}`
	
	`nav a {`
	`  display: block;`
	`  color: white;`
	`}`

De standaard combinator bij gebruik van nesting in SCSS is de descendant combinator (de space combinator) uit CSS. Maar je kan ook andere combinators gebruiken en je kan ook meerdere nesting niveaus hebben. Hieronder vind je een uitgebreid voorbeeld met links in de tabel de SCSS-code en rechts de overeenkomstige gecompileerde code.

SCSS Source code	Compiled CSS code

`nav {`
`  color:#07077c;`
`  ul {`
`  margin: 0;`
`  padding: 0px;`
`  > li {`
`  list-style-type: none;`
`  > a {`
`  display: block;`
`  }`
`  }`
`  }`
`}`
	`nav {`
	`  color: #07077c;`
	`}`
	
	`nav ul {`
	`  margin: 0;`
	`  padding: 0px;`
	`}`
	
	`nav ul > li {`
	`  list-style-type: none;`
	`}`
	
	`nav ul > li > a {`
	`  display: block;`
	`}`

3.2 De 'parent selector'

Indien we in de navigatiebalk de hyperlinks enkel willen onderlijnen als er gehoverd wordt over de links dan kunnen we niet zomaar onze SCSS-code met de a-selector uitbreiden van

```
a {
  display: block;
  color: $link;
}
```

naar

```
a {
  display: block;
  color: $light;
  text-decoration: none;
  :hover {
    text-decoration: underline;
  }
}
```

Denk zelf na waarom dit niet volstaat (bekijk eventueel de door bovenstaande SCSS-code gegenereerde CSS-code).

In dit geval moeten we dus de in SASS ingebouwde parent selector (&) gebruiken.

8. Voeg onderstaande SCSS-code met lichtgroene achtergrond toe aan je .scss-bestand en controleer in **styles.css** dat deze code omgezet wordt naar de CSS-code uit de rechterhelft van onderstaande tabel.

SCSS Source code	**Compiled CSS code**

`nav {`
`  ...`
`  a {`
`  display: block;`
`  color: $light;`
`  text-decoration: none;`
`  &:hover {`
`  text-decoration: underline;`
`  }`
`  }`
`}`
	`nav a {`
	`  display: block;`
	`  color: white;`
	`  text-decoration: none;`
	`}`
	
	`nav a:hover {`
	`  text-decoration: underline;`
	`}`

9. Controleer het resultaat in je browser:

4 Partials

Als je in jouw CSS-code veel CSS `@import`-statements gebruikt dan kan dit het laden van de CSS in de browser vertragen. Meestal zal het performanter zijn als een browser slechts één css-file moet downloaden in plaats van meerdere .css-files.

Partials in combinatie met het SASS `@import`-statement lossen dit probleem op. Partials laten je toe om jouw SCSS-code te verdelen over meerdere files en toch één gecompileerde CSS-file te bekomen.

Een partial is een .scss-file die bedoeld is om geïmporteerd te worden in een andere .scss-file. De bestandsnaam van een partial moet beginnen met een 'underscore'. Zo weet de SASS-compiler dat hij voor deze file geen .css-file moet genereren.

10. Verdeel de SCSS-code in **styles.scss**-file over een aantal partials en importeer vervolgens de partials in **styles.scss**.

    Maak bijvoorbeeld de volgende partials:
    - **_variables.scss**
    - **_basic-styles.scss**
    - **_nav-styles.scss**

    en importeer de partials in **styles.scss**.

styles.scss

```
@import "variables";
@import "basic-styles";
@import "nav-styles";
```

Merk op dat het onderstrepingsteken en de bestandsextensie .scss niet opgenomen zijn in de @import statements. We schrijven dus `@import "variables";` in plaats van `@import "_variables.scss"`.
Controleer of je nog steeds dezelfde **styles.css** file bekomt. Je kan eventueel **styles.css** verwijderen en als je vervolgens **styles.scss** opnieuw bewaart, zal de Live Sass Compiler de **styles.css** opnieuw creëren.