How to: SASS

source: (http://sass-lang.com/)

Indholdsfortegnelse

Hvad er SASS?

Sass er en udvidelse af CSS (Cascading Style Sheet), som giver flere muligheder til standard CSS syntax. SASS inkluderer:

- **Fuld CSS3 kompatibilitet**
- **Udvidelse til CSS syntaxen såsom: variabler, nestings og mixins.**
- **CSS Funktioner**
- **God formatering og ændringsvenlig output.**
- **Nem opdeling af filer**

Hvorfor bruge SASS?

CSS syntax har mangler i forhold til at genbruge kode, som SASS er god til at hjælpe med at gøre automatiseret gennem variabler, nesting og mixins. En anden vigtigt ting, er, at SASS gør det utroligt nemt at strukturere CSS, hvilket er en nødvendighed når der arbejdes i større projekter.

Arbejdes der med CSS på ugentligt basis, kan SASS både spare tid i udviklingsfasen samt rettelsesfasen, da det at skulle udskifte en farve ét sted, gennem en variabel, kontra at udskifte samme farve 50 forskellige steder med copy-pasté, speeder processen med mange længder.

Opsætning

Følg SASS' egen guide til installation (http://sass-lang.com/install).

Når SASS er installeret, skal du oprette en ny folder til dit projekt, som du via. din terminal skal gå ind i. Opret her en fil med endelsen ".scss" (eks: style.scss). Når dette er gjort, kan disse to commands bruges i projektet:

```
sass input.scss output.css
```

Denne command omdanner en .scss fil direkte til css, uden at man selv manuelt skal oprette en css fil.

```
sass --watch input.scss:output.css
```

Denne command gør det samme som den første, men i stedet for at skulle bruge samme command for hver ændring der foretages i .scss filen, holder denne øje med ændringer. Skulle der ske en syntax fejl i .scss filen, vil den derudover også give en error besked samt fortælle hvor i filen / hvilken fil der er error.

SCSS Syntax

I SASS er der to måder at skrive CSS på: SASS og SCSS.
I denne guide vil der følges SCSS standarden, da den læner sig mest op af den måde som CSS skrives på, hvilket ikke gør skiftet mellem de to sprog særlig stor. Dette er dog en præference sag. Her er et eksempel på forskellen mellem de to:

eksempel på *SASS syntax:*

```
nav
  ul
    margin: 0
    padding: 0
    list-style: none

  li
    display: inline-block

  a
    display: block
    padding: 6px 12px
    text-decoration: none
```

eksempel på *SCSS syntax:*

```
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }

  li { display: inline-block; }

  a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
  }
}
```

Som det ses på billederne, så er den store forskel på SASS og almindelig CSS hvordan elementer og properties lukkes. SCSS, derimod, ligner utroligt meget det normale CSS i opbygningen.

Variabler

I SCSS kan der, som tidligere nævnt, bruges variabler. Variablerne virker præcis ligesom i php, javascript eller hvilket som helst andet programmeringssprog. Det indeholder data, som kan genbruges flere gange. I SCSS kan variabler blandt andet bruges til at indeholde en farve:

```
1  $white: #fff;
2  $black: #000;
3  $blue: #45657e;
4  $darkBlue: #34495E;
5  $green: #2ecc71;
6  $dimgray: dimgray;
7
```

Når variablerne derefter skal bruges, bliver de kaldt således:

```
1  header {
2      height: 120px;
3      background: $white;
4      width: 100%;
5      z-index: 9999;
6      position: relative;
```

Her ses det, at den hvide baggrundsfarve bliver brugt i headeren. Når den bliver outputtet til CSS, vil den automatisk skifte **$white** ud med **#fff**.

Variabler kan bruges til rigtigt mange usecases, og giver en dejlig hurtig måde at lave ting som farvekoder og borders på, så man ikke skal huske dem i hovedet, men bare kalde dem gennem variablen hver gang.

Nesting

Når man skriver HTML, er der en klar visuel struktur over, hvilke elementer der hører sammen med hvad. I CSS findes der ikke denne form for struktur. I SASS er dette dog muligt. Her er et eksempel på HTML kode for en header:

```
<header>
    <nav>
        <ul>
            <li><a href="#">Menu item 1</a></li>
            <li><a href="#">Menu item 2</a></li>
            <li><a href="#">Menu item 3</a></li>
            <li><a href="#">Menu item 4</a></li>
            <li><a href="#">Menu item 5</a></li>
        </ul>
    </nav>
</header>
```

Her er SCSS syntaxen for at style headeren:

```scss
header {
    height: 50px;
    width: 1200px;

    nav {
        width: 50%;
        height: 100%;
        margin: 0 auto;

        ul {
            float: left;

            li {
                width: 50px;
                height: 100%;

                a {
                    text-decoration: none;

                } /* ends a-tag */
            } /* ends li-tag */
        } /* ends ul-tag */
    } /* ends nav-tag */
} /* ends header-tag */
```

Som der ses på dette eksempel, skriver man nav videre i headeren, ul videre i nav osv. Dette følger html strukturen væsentligt bedre, og giver en meget bedre visuel struktur på ens kode. Når SCSS ryger igennem preprocessoren, vil det se således ud:

```scss
header {
    height: 50px;
    width: 1200px;
}

header nav {
    width: 50%;
    height: 100%;
    margin: 0 auto;
}

header nav ul {
    float: left;
}
```

Media queries

Udover at kunne neste elementer i hinanden i SCSS, kan media queries *også* nestes i de forskellige elementer. Et eksempel kunne være, at navigationen skal blive 100% bred under 1200 pixels bredde:

```
5    nav {
6        width: 50%;
7        height: 100%;
8        margin: 0 auto;
9        @media screen and (max-width: 1200px) {
10           width: 100%;
11       }
12
```

Her ses det, at man kan skrive media queries direkte inde i elementet. Dette ville lave et output der ser således ud (her tages der højde for, at nav er inde i header).:

```
35  @media screen and (max-width: 1200px) {
36      header nav {
37          width: 100%;
38      }
39  }
```

Mixins

I CSS3 er der tit behov for at prefixe forskellige CSS properties. Her kan det være utroligt tidskrævende, hvis ikke man har en god metode at gøre det med. I SASS kan man bruge mixins, som lader en lave en gruppe af properties, som kan genbruges flere gange i koden. Et eksempel på en mixin:

```
@mixin border-radius($radius) {
  -webkit-border-radius: $radius;
     -moz-border-radius: $radius;
      -ms-border-radius: $radius;
          border-radius: $radius;
}
```

Hver gang at der skal laves en border radius, kan det blive gjort ved at skrive følgende:

```
.box { @include border-radius(10px); }
```

Det samme kunne gøres ved en masse andre ting, hvilket gør det hurtigere og mere automatiseret at bruge disse prefixes.

Extend

I CSS kan man komme ud for, at skulle give flere elementer, de samme properties. I SASS kan man gøre dette ved at bruge **@extend**. Et eksempel kunne være dette:

```
.message, .success, .error, .warning {
  border: 1px solid #cccccc;
  padding: 10px;
  color: #333;
}

.success {
  border-color: green;
}

.error {
  border-color: red;
}

.warning {
  border-color: yellow;
}
```

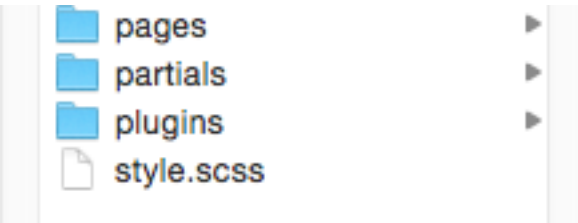
I stedet for at skrive elementerne flere gange, kan man bruge extend:

```
.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;
}
```

Extend tager her alle properties med, fra det elementer man extender.

Projektstrukturering med SASS

I SASS er det muligt, at opdele CSS kode i mindre filer. Dette er fantastisk i forhold til at holde styr på de forskellige dele i ens kode, samt senere at vedligeholde CSS'en. Arbejder man med et større projekt, kan dette dog give problemer, hvis der ikke er lavet en god mappestruktur til projektet, da mange filer hurtigt kan give den modsatte effekt, ved at blive uoverskueligt.
En struktur kunne være følgende:



I denne struktur er **style.scss** filen hovedfilen hvorfra, alle andre filer bliver hentet ind. De andre filer er inddelt i 3 mapper: **Pages**, **Partials** og **Plugins**. Måden de forskellige sider bliver hentet ind på, er ved hjælp af **@import.**

Et eksempel på en style.scss fil, hvor der bliver brugt import:

```
40  /*=================================
41  =              PAGES              =
42  =================================*/
43
44  /* Front Page */
45  @import "pages/frontpage/cover";
46  @import "pages/frontpage/slider";
47  @import "pages/frontpage/question-help";
48  @import "pages/frontpage/userstory";
49  @import "pages/frontpage/quote-slider";
50
51  /* Single Page*/
52  @import "pages/single/coversingle";
53  @import "pages/single/Content";
54  @import "pages/single/customwrap";
55  @import "pages/single/single-news";
56  @import "pages/single/inspirations-single";
57  @import "pages/single/single-refs";
58  @import "pages/single/single-emner";
59
60  /* Foredrag and Emner */
61  @import "pages/foredragsholdere-emner/filter-bar";
62  @import "pages/foredragsholdere-emner/letter-sections";
63  @import "pages/emner/emner";
64
```

Som der ses på billedet, skal importen lede direkte hen til filen der skal importeres. Derfra lavet den en copy-pasté af koden, og samler outputtet i én fil (style.css). Når der laves import, skal der ikke skrives fil endelser på, da SASS ved, at den skal kigge efter **.scss** endelserne.

Pages

I denne mappe skal alle siderne til projektet være. Hvis det er et lille projekt, kan det være nok bare med denne mappe, men ellers rådes der til, at lave undermapper til hver enkelt side. Hvis der kigges på billedet før, så er der f.eks lavet en mappe til **forsiden(frontpage)**. I denne mappe ligger

der en fil for alle de sektioner der er på forsiden. Dette giver en rigtig god fil-struktur, da det er nemt at vide hvor stylen til de forskellige sektioner findes henne.

Partials

I denne mappe bør være to undermapper: **Assets** og **Shared** (eller hvad man finder mest passende i forhold til navngivning). I mappen **Assets** kan man have filer såsom variabler med farvekoder og mixins. I **Shared** bør alle de partials der bliver delt mellem de forskellige sider på websitet være. Det kan f.eks være navigationen, footeren, newsletter signup osv. Ting der bliver genbrugt flere gange, og som ikke kun hører med under én enkelt side.

Plugins

I denne mappe bør alle plugins der bruges være. Det kan være css reset, normalize, clearfix eller noget helt andet. Oftest er det filer, som er statiske, det vil sige, filer som man ikke retter i løbende. Disse bør også importeres som det første i style.scss, da resten af ens kode kan være afhængig af disse.