

CSS PREPROCESSORY

Co jsou to CSS preprocesory a k čemu se používají.

SASS

Syntactically Awesome Style Sheets je preprocesorový skriptovací jazyk, který dovoluje zkompilevat zdrojové kódy do použitelného CSS. Používá dva formáty a to SCSS a SASS, kde SCSS je velmi podobné CSS (a dokonce CSS je správný SCSS syntax) a SASS je bez složených závorek a je postaveno na indentaci (např. stejně, jako Python). V tomto předmětu budeme používat zejména SCSS. Často se říká, že SASS je metajazyk, tedy, že je to jazyk o jazyku.

SASS vypadá následovně:

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

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

SCSS vypadá následovně:

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

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

Důvod existence SASSu je spousta. Nejdůležitější je to, že SASS dovoluje tvořit přehlednější zápis, a ještě nám zrychluje práci (např. proměnnými a vnořováním). V kódu můžeme taky tvořit funkce a tzv. mixiny a znovu používat styly.

Zdrojové kódy mají přípony `.scss` a `.sass`. Zdrojové kódy pomocí preprocesoru zkompilejeme do použitelného CSS. Jedním z kompilátorů je sass v npm, který je napsán v Dartu. Setkáte se například ale i s node-sass a dalšími. Všechny preprocesory se mohou lišit svým chováním (a nějaké věci nemusí vůbec mít implementované). Můžeme kompilovat např. i na webu [ZDE](#).

Knihovnu sass globálně nainstalujeme v npm pomocí příkazu:

```
npm install -g sass
```

Poté můžeme v příkazové řádce použít příkaz:

```
sass --version
```

Který nám vypíše verzi SASS.

Kompilovat můžeme jedním příkazem, které má různé přepínače:

- `--no-source-map`, který zamezí generování map zdrojů pro jednotlivé řádky (v DevTools neuvidíme, kde se v SASS daná věc vzala)

- `--watch` , který bude daný soubor (a jeho vázané soubory) sledovat a v případě změn je automaticky přegeneruje

```
sass source.scss target.css  
sass --no-source-map source.scss target.css  
sass --watch source.scss target.css
```

Dokumentaci SASSu nalezneme [ZDE](#).

Poznámky:

Žádné poznámky k této lekci jste ještě nezapsali.

ZLEPŠENÍ V SASS

Vnořování

Při tvorbě stylů často potřebujeme různě vnořovat jednotlivé selektory do sebe. V obyčejném CSS musíme pravidla rozdělovat a často se stane, že vlastně není viditelné, jak jednotlivé vnořování funguje. V SASSu máme možnost psát selektory do selektorů. Když to vytvoříme, tak se dle daných selektorů vygenerují pravidla separovaně.

```
header {
  display: flex;
  justify-content: center;

  nav {
    min-width: 70vh;

    > ul {
      display: flex;
      gap: 10px;

      > li a {
        color: black;
        text-decoration: underline;
      }

      a.special {
        color: rebeccapurple;
      }
    }
  }
}
```

Se zkompiluje na:

```
header {
  display: flex;
  justify-content: center;
}
header nav {
  min-width: 70vh;
}
header nav > ul {
  display: flex;
  gap: 10px;
}
header nav > ul > li a {
  color: black;
  text-decoration: underline;
}
header nav > ul a.special {
  color: rebeccapurple;
}
```

Proměnné

Proměnné mohou být v SASSu jakéhokoliv datové typu. Proměnnou definujeme vždy pomocí dolaru a musíme určit hodnotu. Na dalších místech ji referujeme s dolarem a jejím jménem. SASS nekontroluje zda daná hodnota může do daného vlastnosti přijít, prostě ji tam vloží:

```
$primaryColor: #37e2e2;
```

```
body > main {
```

```

    section#hero {
      background-color: $primaryColor;
      font-weight: $primaryColor;
    }

    a.link.special {
      color: $primaryColor;
    }
  }
}

```

Se zkompiluje na:

```

body > main section#hero {
  background-color: #37e2e2;
  font-weight: #37e2e2;
}
body > main a.link.special {
  color: #37e2e2;
}

```

S proměnnými můžeme dále pracovat:

```

$fontFamily: 'Comis Sans', 'Comis Sans MS', sans-serif;
$size: 100px;

body {
  font-family: $fontFamily;

  h1 {
    font-size: $size;
  }

  h2 {
    font-size: $size / 1.5;
    margin-top: $size * 4;
    margin-bottom: $size - 30px;
  }
}

```

Se zkompiluje na:

```

body {
  font-family: "Comis Sans", "Comis Sans MS", sans-serif;
}
body h1 {
  font-size: 100px;
}
body h2 {
  font-size: 66.6666666667px;
  margin-top: 400px;
  margin-bottom: 70px;
}

```

Kód reálné nepůjde zkompilovat, protože operátor dělení je aktuálně nepodporovaný a je nutné použít knihovnu:

```

@use "sass:math"; // načti knihovnu
$size: 100px;

body {
  h2 {
    font-size: math.div($size, 1.5); // ==== $size / 1.5 ==== 66.6666666667px
  }
}

```

```
}  
}
```

Aktuální prvek

Při vnořování často potřebujeme aktuálnímu prvku (tj. tomu selektoru, ve kterém se nacházíme - rodiči) přidat něco na více. Proto existuje vybrání aktuálního prvku pomocí operátoru `&`. Například tedy:

```
$textOnPrimary: white;  
$text: #141414;  
  
.button {  
  display: flexbox;  
  align-items: center;  
  justify-content: center;  
  
  gap: 10px;  
  
  background-color: #aaaaaa;  
  
  &.special {  
    background-color: $primaryColor;  
    color: $textOnPrimary;  
  }  
  
  &:hover { // Zkuste si např. zde & smazat a porovnejte výsledek s tím níže!  
    background-color: #b6b6b6;  
  
    &.special {  
      background-color: $primaryColorInteract;  
    }  
  }  
}
```

Se zkompiluje na:

```
.button {  
  display: flexbox;  
  align-items: center;  
  justify-content: center;  
  gap: 10px;  
  background-color: #aaaaaa;  
}  
.button.special {  
  background-color: rgba(224, 134, 82, 0.8);  
  color: white;  
}  
.button:hover {  
  background-color: #b6b6b6;  
}  
.button:hover.special {  
  background-color: rgba(252, 184, 144, 0.8);  
}
```

Práce s barvami

V SASS máme předpřipravených pár funkcí na práci s barvami. Zejména se jedná o úpravy jejich vzhledu. Všechny naleznete [ZDE](#).

```
$primaryColor: #8df0ae;
```

```
a.active {
```

```
color: darken($primaryColor, 10%); // Snižuje světlost pevně
color: color.scale($primaryColor, $lightness: -10%); // Snižuje světlost o procenta
}
```

Se zkompiluje na:

```
a.active {
  color: #60ea8e;
  color: #6bec96;
}
```

Moduly

V SASS můžeme různě importovat soubory do sebe. Často se nám hodí importovat i do vnořování. Používáme `@import` (zastaralé, ničí "scope" proměnných a dalších) a nebo `@use` .

```
@import "./variables";

main {
  > {
    @import "./homepage";
  }
}
@import "./buttons";
```

`@use` však nelze používat ve vnořování a musíme používat tzv. mixiny a přistupuje dle importovaného jména.

Komentáře

Jak jste si již všimli, v SASS můžeme používat řádkové komentáře pomocí `//` . Všechny komentáře se v kompilaci smažou.

Poznámky:

Žádné poznámky k této lekci jste ještě nezapsali.

KOMPILACE

Při chybě v kompilaci se do cílového souboru předá textový výstup, který bude zobrazitelný na stránce pomocí pseudotříd. V konzoli kompilátoru tyto chyby nalezneme taktéž. Chybové a varovné hlášky jsou velmi přehledné.

Poznámky:

Žádné poznámky k této lekci jste ještě nezapsali.

FUNKCE A MIXINY

Mixiny

Mixiny si můžeme představit jako funkce např. v JS. V SASS jsou funkce však něco jiného (viz. níže). Mixiny dovolují různé styly vkládat do jiných stylů.

```
@mixin reset-list {
  margin: 0;
  padding: 0;
  list-style: none;
}

@mixin horizontal-list {
  @include reset-list;

  li {
    display: inline-block;
    margin: {
      left: -2px;
      right: 2em;
    }
  }
}

nav ul {
  @include horizontal-list;
}

nav ol {
  @include horizontal-list;
}
```

Se zkompiluje na:

```
nav ul {
  margin: 0;
  padding: 0;
  list-style: none;
}
nav ul li {
  display: inline-block;
  margin-left: -2px;
  margin-right: 2em;
}

nav ol {
  margin: 0;
  padding: 0;
  list-style: none;
}
nav ol li {
  display: inline-block;
  margin-left: -2px;
  margin-right: 2em;
}
```

Při volání můžeme také předávat parametry:

```
@mixin flex($justifyContent: start, $alignItems: start, $gap: 15px) {
  display: flex;
  justify-content: $justifyContent;
  align-items: $alignItems;
}
```

```

    row-gap: $gap;

    @if $gap > 10px {
        column-gap: $gap * 1.5;
    } @else {
        column-gap: $gap + 5px;
    }
}

.products {
    @include flex();

    > .product {
        @include flex(center, stretch, 5px);
    }
}

```

Se zkompiluje na:

```

.products {
    display: flex;
    justify-content: start;
    align-items: start;
    row-gap: 15px;
    column-gap: 22.5px;
}
.products > .product {
    display: flex;
    justify-content: center;
    align-items: stretch;
    row-gap: 5px;
    column-gap: 10px;
}

```

Funkce

Funkce na rozdíl od mixins, které vrací vlastnosti či pravidla, vrací jednu hodnotu, která lze využít v hodnotách a dalších místech. Hodí se tedy zejména na různé počítání a nastavení. V níže uvedené ukázce používáme podmínky, které si vysvětlíme v další stránce.

```

@function coloring($priority) {
    $value: #ffffff;
    @if $priority == 1 {
        $value: #cccccc;
    } @else if $priority == 2 {
        $value: #25d189;
    }

    @return $value;
}

body {
    background-color: coloring(1);
    .warning {
        background-color: coloring(2);
    }

    .awesome {
        background-color: coloring(0);
    }
}

```

Se zkompiluje na:

```
body {  
  background-color: #cccccc;  
}  
body .warning {  
  background-color: #25d189;  
}  
body .awesome {  
  background-color: #ffffff;  
}
```

Poznámky:

Žádné poznámky k této lekci jste ještě nezapsali.

CYKLY A PODMÍNKY

Podmínky v SASSu řešíme pomocí pravidel `@if` , `@else` a `@else if` , které jasně popisují co dělají. Tyto pravidla můžeme používat uvnitř pravidel:

```
.test {
  @if true {
    color: green;
  } @else {
    color: red;
  }
}
```

`@else` lze použít max. jednou a to za blokem `@if` nebo `@else if` . Blok `@else if` musí být použit za `@if` a nebo `@else if` .

Cykly se v SASS tvoří pomocí pravidla `@each` :

```
@each $size in 50, 30, 10 {
  .padding-#{$size} {
    padding: #{$size}px;
  }
}
```

Se zkompiluje na:

```
.padding-50 {
  padding: 50px;
}

.padding-30 {
  padding: 30px;
}

.padding-10 {
  padding: 10px;
}
```

SASS obsahuje taktéž `@for` (dokumentace [ZDE](#)) a `@while` (dokumentace [ZDE](#)), které jsou skoro stejným přepisem jako např. v programovacím jazyce C# či JS.

Poznámky:

Žádné poznámky k této lekci jste ještě nezapsali.

LESS

LESS je méně známá alternativa SASSu. LESS je obecně méně přehlednější a používá kompilátor napsán v JS. Více informací o LESSu naleznete [ZDE](#).

Poznámky:

Žádné poznámky k této lekci jste ještě nezapsali.

