CSS preprocesory

SSPŠ
MVOP3 WBI
2025 / 2026
Ing. Matěj Cajthaml

Moderní CSS

- Nejvíce hýbe s webem
- HTML a core "JS" v podstatě již konstantní okolo 3 let

- Existuje spousta nových funkcionalit, o kterých nejspíše nevíte
- Nyní si představíme ty, které již podporuje řada prohlížečů:

- Uživatelské proměnné
- Vnořování

- Mezi další se řadí:
- Uživatelské funkce
- Uživatelské mixiny
- Vylepšené operátory pro barvy
- Container queries (skvělé, ale mimo rámec)

Proměnné

- Dovolují definovat opakující hodnotu
- Ta se může v průběhu CSS pravidel přepisovat
- Navíce je můžeme upravovat pomocí DOM
- A hodnota se propíše

- Jeden nový selektor
- :root => vybírá "kořen celého html"
- Což může být buď html, nebo např. svg
- Navíc má větší specificitu

- Jedná nová funkce, var

- Proměnné jsou vlastně vlastní vlastnosti

Funkce var

- Pokusí se najít hodnotu proměnné ve stromě
- Resp. snažíme se najít proměnnou daného jména na aktuálním prvku
- Bereme "preferenční" (specifičtější) selektory, stejně jako u vlastností
- Pokud nenajde, hledáme v rodiči, v rodiči...
- Prvním argumentem je název proměnné
- Druhým je hodnota, která se použije, pokud není hodnota nalezena, nepovinné

Použití / 1

```
:root {
    --primary-color: #fdee3f;
}
header div.container .logo {
    background-color: var(--primary-color);
}
```

Použití / 2

```
1 :root {
2     --primary-colora: #fdee3f;
3 }
4 header div.container .logo {
5     background-color: var(--primary-color, #ff00ff);
6 }
```

Použití / 3

```
:root {
    --primary-color: #fdee3f;
}

header div.container {
    --primary-color: #fefefe;
}

header div.container .logo {
    background-color: var(--primary-color, #ff00ff);
}
```

Použití / 4

```
:root {
    --logo: url("./logo.png");
}

header div.container .logo {
    background-image: var(--logo, url("/img/logo.png"));
}
```

OTÁZKA

Napadá vás, kde by proměnné nešlo využít?

Nefunkční

```
:root {
    --logo: "./logo.png";
}

header div.container .logo {
     background-image: url(var(--logo, "/img/logo.png"))
}
```

Vnořování

- Psaní selektorů v CSS je náročné
- Nejen o pochopení, ale zejména přehlednost
- Velký "skok" pro lidi, kteří se to chtějí naučit

- V moderní CSS je možné do těla pravidla vložit jiné celé pravidlo, které svůj selektor "znásobí" s tím nadřazeným
- Intuitivní a přehlednější pro všechny

- Nový operátor pro selektory
- & - "pokračování z rodiče"

- IDE nejsou moc připravená

- V prohlížeči se rovnou interpretuje, nepřekládá

Vnořování / 1

```
.article {
    border: 1px solid black;

    .title {
        font-size: 2rem;
    }
}
```

```
.article {
    border: 1px solid black;
}

.article .title {
    font-size: 2rem;
}
```

Vnořování / 2

```
.article {
    border: 1px solid black;

    .title {
        font-size: 2rem;

        + .subheading {
            margin-top: 0.25em;
        }

        > p {
            font-size: 1.3em;
        }
    }

```

```
.article {
    border: 1px solid black;
}

.article .title {
    font-size: 2rem;
}

.article .title + .subheading {
    margin-top: 0.25em;
}

.article .title > p {
    font-size: 1.3em;
}
```

Selektor vnoření

- &
- Lze si představit jako "zástupce" rodičovského selektoru
- Řeší problém, když chceme daný selektor modifikovat přímo během vnořování
- Znovu zejména pro přehlednější kód

Vnořování / 3

```
.button {
    background-color: blue;

    &:hover {
        background-color: red;
    }
}
```

```
.button {
    background-color: blue;
}

.button:hover {
    background-color: red;
}
```

Vnořování / 4

```
.container {
    width: 100%;

    @media (min-width: 768px) {
        width: 750px;
    }
}
```

```
.container {
    width: 100%;
}

@media (min-width: 768px) {
    .container {
        width: 750px;
    }
}
```

Preprocesory

Aneb jak předpracujeme CSS

Preprocesor

- Velmi podobné s metajazyky
- Viděli jsme již s TS
- Pracuje nad setem nějakého kódu a vrací kód upravený a použitelný někde jinde
- Pro CSS existuje nespočet metajazyků:
- **Sass** (nejznámější)
- Less
- Stylus
- Proč vůbec vznikli a historie webu
- Použití i v moderním webu

Sass

- Dlouhá historie
- CSS vzala spoustu nápadů
- Syntaxe navazuje na klasické CSS

- Dva formáty:
- **SCSS** - vypadá jako CSS, vlastní zápisy
- Sass (ano...) - vypadá jako Python, odstraňuje složené závorky, středníky a další

- Přípony jsou .scss a .sass

- Kompilace
- Spousta nástrojů
- Balíček v npm sass

- npx sass input.scss output.css
- Přepínač --watch

Proměnné

- V podstatě stejné jako v CSS
- Velký problém: ve výsledném CSS nejsou vidět
- A tedy nejdou ani za běhu měnit
- Zapisujeme pomocí znaku dolaru
- Použití pouze referencí na jméno

- Lze je používat např. v @media

Sass / 1

```
$primary-color: #3498db;

.button {
    background-color: $primary-color;
}
```

```
.button {
    background-color: #3498db;
}
```

OTÁZKA

Můžeme definovat CSS proměnné uvnitř Sass kódu?

Vnořování

- Ekvivalentní k (aktuálnímu) CSS
- Stejný selektor &
- Jde ale dělat víc "cool" věci s rodičovským selektorem

Sass / 2

```scss
$primary-color: #3498db;

.button {
    background-color: white;

    &.primary {
        background-color: $primary-col
    }
    &--icon {
        font-size: 3em;
    }
}
```

```scss
.button {
  background-color: white;
}
.button.primary {
  background-color: #3498db;
}
.button--icon {
  font-size: 3em;
}
```

Základní konstrukce

- Podmínky
- @if
- @else
- @else if

- Cykly
- @for
- @each
- a další

Sass / 3

```
$theme: dark;
.button {
    @if $theme == dark {
        background-color: black;
        color: white;
    } @else {
        background-color: white;
        color: black;
    }
}
```

```
.button {
    background-color: black;
    color: white;
}
```

Sass / 4

```scss
@for $i from 1 through 3 {
    .col-#{$i} {
        width: 33.333% * $i;
    }
}
```

```css
.col-1 {
    width: 33.333%;
}

.col-2 {
    width: 66.666%;
}

.col-3 {
    width: 99.999%;
}
```

Sass / 5

```scss
@for $i from 1 through 12 {
    .col-#{$i} {
        width: (100% / 12) * $i;
    }
}
```

```
Using / for division outside of calc()

Recommendation: math.div(100%, 12) or c



More info and automated migrator:

https://sass-lang.com/d/slash-div
```

Sass / 5

```scss
@use "sass:math";
@for $i from 1 through 12 {
    .col-#{$i} {
        width: math.div(100%, 12) * $i;
    }
}
```

```css
.col-1 {
  width: 8.3333333333%;
}

.col-2 {
  width: 16.6666666667%;
}

.col-3 {
  width: 25%;
}

.col-4 {
  width: 33.3333333333%;
}
```

Sass / 6

```
$base-padding: 10px;
.container {
    padding: $base-padding * 2;
}
```

```
.container {
    padding: 20px;
}
```

Operátory a typy

- Základní aritmetické operace
- Pozor např. na /, viz dříve

- Řetězce
- Boolean
- Čísla - bez jednotek a s nimi
- Barvy
- Seznamy
- Mapy

Sass / 7

```
$font-stack:    Helvetica, sans-serif;
$primary-color: #333;
$base-margin: 16px;
$is-enabled: true;

$theme-colors: (
  "primary": #3498db,
  "secondary": #2ecc71,
  "danger": #e74c3c
);

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

Sass / 8

```
$colors: ("red", "green", "blue");
@each $color in $colors {
    .text-#{$color} {
      color: $color;
    }
}
```

Moduly

- Již jsme viděli dříve (sass:math)

- Dovoluje nám kód rozdělit do více souborů
- Např. proměnné někam, jednotlivé komponenty do souborů

- Máme dvě pravidla
- @import, zastaralé, nepoužívat
- @use

- Jako parametr dáváme název souboru
- Uloží se pod klíčem jména daného souboru
- Či můžeme přejmenovat pomocí operátoru as

Sass / 9

```
@use 'variables' as vars;
.button {
    background-color: vars.$primary-color;
}
```

Mixiny a funkce

- Funkce -> přijímá vstup, vrací hodnotu jako výstup
- Např. vypočítání barvy, hodnoty vůči jiné, ...

- Mixin -> přijímá vstup, může vracet CSS vlastnosti a pravidla
- Např. vložení kódu, který se stále opakuje, např. flex, specifické styly

- Oboje dovoluje rozšiřitelnost v budoucnosti
- @return, @function, @mixin
- @include

Sass / 10

```scss
@use "sass:math";

@function fluid-text($min-px, $max-px, $min-vw: 320, $max-vw: 1200) {
  $slope: math.div($max-px - $min-px, $max-vw - $min-vw);

  $y-axis-intersection: -$min-vw * $slope + $min-px;

  $min-rem: math.div($min-px, 16);
  $max-rem: math.div($max-px, 16);
  $intercept-rem: math.div($y-axis-intersection, 16);
  $slope-vw: $slope * 100;

  @return clamp(
    #{$min-rem}rem,
    #{$intercept-rem}rem + #{$slope-vw}vw
```

Sass / 11

```scss
@mixin gradient-border($border-width, $gradient, $radius: 0) {
  position: relative;
  border-radius: $radius;
  background-clip: padding-box;
  border: none;

  &::before {
    content: '';
    position: absolute;

    inset: -#{$border-width};

    z-index: -1;

    background: $gradient;
```

Sass / 12

```
@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;
    }
  }
```

Alternativy a budoucnost

Aneb jak to dělat jinak a možná vůbec?

Alternativy

- Již zmíněné
- Stylus
- Less
- A další

- Obecně nedělají nic jiného než SASS
- Rozdíl zejména v syntaxi
- Možné "developerské nástroje", které jsou k dispozici

- Co jiného mají?

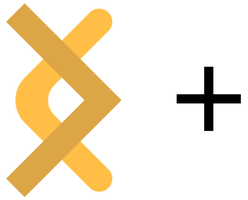
OTÁZKA

Co dále byste rádi viděli v CSS?

Budoucnost CSS

- Chystá se toho hodně
- Tvorba "pozdě"
- Specifikace, standard aby nebyly chyby (viz 10 dnů s JS)
- Externí nástroje pomáhají s tvorbou, kompetice
- Jinak stagnace

Děkuji za pozornost



© 2025 Matěj Cajthaml