Introduction do SASS

Outline

- SASS – basics
- Variables
- Mixin
- Extend
- Functions
- if, else
- Loops – each, for, while
- Math features

What is SASS?

- SASS (*Syntactical Awesome Stylesheet*) is a scripting language whose code is processed into the resultfiles of cascading style sheets - CSS.
- Two encoding syntax:
  - Sassy CSS (.scss) – from CSS
  - from scripting language HAML, where buckles and semicolons are omitted (.sass)

Installation of SASS

- http://rubyinstaller.org/downloads/
- Wiersz poleceń z obsługą Ruby - **Start Command Prompt with Ruby** (dla Windows)
- Installation by nam:
  `npm install node-sass -g`
- `node-sass –v`
- Transcompilation (transpilation)
  `node-sass plik.scss:plik.css`
- `It can be also enabled in IDE like (VS Code or Atom)`

SASS - comments

- SASS adds the one-line comment option, but they are not visiblew plikach .css

main.scss

```
// invisible comment
// in *.css


/* visible comment in
   *.css
*/
```

main.css

```
/*
   visible comment in
   *.css
*/
```

SASS – imports

- CSS rarely uses `@import`
- `@import in .scss and .sass is executed during compilation and saved to a single .css file`
- Adding file extensions are optional:
- `@import "buttons";`
- `Especially useful in the case of code separation and repeated use of its parts`

Nesting

main.scss

```
.content {
    font-size: 12px;
    color: green;
    p {
        margin: 15px 0;
    }
    h1 {
        margin: 30px 15px;
        border: 2px solid red;
    }
}
```

main.css

```
.content {
    font-size: 12px;
    color: green;
}
.content p {
    margin: 15px 0;
}
.content h1 {
    margin: 30px 15px;
    border: 2px solid red;
}
```

Parent Selector

- During nesting, you can use the & operator, which is responsible for the parent selector.

main.scss

```
.content {
   font-size: 12px;
   p {
      margin: 15px 0;
   }
   h1 {
      margin: 30px 15px;
   }
   .callout {
      color: red;
   }
   &.callout {
      color: green;
   }
}
```

main.css

```
.content {
   font-size: 12px;
}
.content p {
   margin: 15px 0;
}
.content h1 {
   margin: 30px 15px;
}
.content .callout {
   color: red;
}
.content.callout {
   color: red;
}
```

Parent Selector

- Very often used in conjunction with pseudo classes.

main.scss

```
a {
    color: blue;
    &:hover {
        color: red;
    }
    &:active {
        color: green;
    }
}
```

main.css

```
a {
    color: blue;
}
a:hover {
    color: red;
}
a:active {
    color: green;
}
```

Parent Selector

- Selectors can also be added before &.

main.scss

```
.contact {
    float: left;
    width: 300px;
    .footer & {
        width: 400px;
    }
}
```

main.css

```
.contact {
    float: left;
    width: 300px;
}
.footer .contact {
    width: 400px;
}
```

Multilevel nesting with Paren Selector

main.scss

```
.content {
   color: blue;
   .callout {
      h2 {
         a {
            &:hover {
               color: red;
            }
         }
      }
   }
}
```

main.css

```
.content {
   color: blue;
}
.content .callout h2 a:hover {
   color: red;
}
```

Variables

- We declare variables in SASS using the $ tag, e.g. $variable.

main.scss

```
$color: #232323;

.contact {
   border: 1px solid $color;
   li {
      color: $color;
   }
}
```

main.css

```
.contact {
   border: 1px solid #232323;
}
.contact li {
   color: #232323;
}
```

Variable types

- Boolean

```
$radius: false;
$shadow: true;
```

- Numbers – no need to provide units

```
$font-size: 1.5em;
$line-height: 1.2;
$border: 3px;
```

Variable types

- Colors

```
$color: red;
$border: #rgba(0, 255, 0, 0.5);
$shadow: #333;
```

- Strings – can be declared with or without „”/,’

```
$header: ‘Helvetica’;
$font-family: Arial;
$message: ”Loading...”;
```

Variable types

- Lists

```
$authors: pawel, mirek, andrzej, krzysztof;
$margin: 30px 0 20px 80px;
```

Variable - scope

- Variables set inside declarations (between {}) cannot be used outside this block!

main.scss

```
p {
    $color: #ccc;
    border: 2px solid $color;
}
h1 {
    border: 2px solid $color;
}
```

main.css

```
Syntax error: Undefined
variable: „$color”.
```

Variable - scope

- By setting new values for variables declared outside the declaration block, it changes the value permanently.

main.scss

```
$color: #232323;

.contact {
    $color: #555555;
    background: $color;
}
h1 {
    color: $color;
}
```

main.css

```
.contact {
    background: #555555;
}
h1 {
    color: #555555;
}
```

Interpolation of variables

- Using the # {$variable} tag, we can use variables in selectors, property names, or strings.

main.scss

```
$side: top;

body {
    position: relative;
    #{$side}: -0.5em;
}
.callout-#{$side} {
    background: blue;
}
```

main.css

```
body {
    position: relative;
    top: -0.5em;
}
.callout-top {
    background: blue;
}
```

Mixins

main.css

```
.btn-a {
   background: #777;
   border: 1px solid #ccc;
   font-size: 1em;
   text-transform: uppercase;
}
.btn-b {
   background: #ff0;
   border: 1px solid #ccc;
   font-size: 1em;
   text-transform: uppercase;
}
```

Compare these two declarations. Common parts can be declared with mixing.

Mixin - declaration

main.scss

```
@mixin button {
   border: 1px solid #ccc;
   font-size: 1em;
   text-transform: uppercase;
}
.btn-a {
   @include button;
   background: #777;
}
.btn-b {
   @include button;
   backgroudn: #ff0;
}
```

main.css

```
.btn-a {
   border: 1px solid #ccc;
   font-size: 1em;
   text-transform: uppercase;
   background: #777;
}
.btn-b {
   border: 1px solid #ccc;
   font-size: 1em;
   text-transform: uppercase;
   background: #ff0;
}
```

Mixin – arguments

main.scss

```
@mixin box-sizing($x) {
   -webkit-box-sizing: $x;
   -mox-box-sizing: $x;
   box-sizing: $x;
}
.content {
   @include box-sizing(border-box);
   border: 1px solid #ccc;
   padding: 20px;
}
.callout {
   @include box-sizing(content-box);
}
```

main.css

```
.content {
   -webkit-box-sizing: border-box;
   -moz-box-sizing: border-box;
   box-sizing: border-box;
   border: 1px solid #ccc;
   padding: 20px;
}
.callout {
   -webkit-box-sizing: content-box;
   -moz-box-sizing: content-box;
   box-sizing: content-box;

}
```

Mixin – arguments with default value

main.scss

```
@mixin box-sizing($x: border-box) {
   -webkit-box-sizing: $x;
   -mox-box-sizing: $x;
   box-sizing: $x;
}
.content {
   @include box-sizing;
   border: 1px solid #ccc;
   padding: 20px;
}
.callout {
   @include box-sizing(content-box);
}
```

main.css

```
.content {
   -webkit-box-sizing: border-box;
   -moz-box-sizing: border-box;
   box-sizing: border-box;
   border: 1px solid #ccc;
   padding: 20px;
}
.callout {
   -webkit-box-sizing: content-box;
   -moz-box-sizing: content-box;
   box-sizing: content-box;

}
```

Mixin – more arguments

main.scss

```
@mixin button($radius, $color)
{
   border-radius: $radius;
   color: $color;
}
.btn-a {
   @include button(4px, #000);
}
```

main.css

```
.btn-a {
   border-radius: 4px;
   color: #000;
}
```

Mixin – more arguments

main.scss

```
@mixin button($radius, $color: #000)
{
   border-radius: $radius;
   color: $color;
}
.btn-a {
   @include button(4px);
}
```

main.css

```
.btn-a {
   border-radius: 4px;
   color: #000;
}
```

Mixin – interpolation

main.scss

```
@mixin highlight($color, $side) {
   border-#{$side}-color: $color;
}
.btn-a {
   @include highlight(#ff0, right);
}
```

main.css

```
.btn-a {
   border-right-color: #ff0
}
```

Extend

main.scss

```
.btn-a {
   background: #777;
   border: 1px solid #ccc;
   font-size: 1em;
   text-transform: uppercase;
}
.btn—b {
   @extend btn-a;
   background: #ff0;
}
```

main.css

```
.btn-a,
.btn-b {
   background: #777;
   border: 1px solid #ccc;
   font-size: 1em;
   text-transform: uppercase;
}
.btn—b {
   background: #ff0;
}
```

Nesting with Extend

main.scss

```
.content {
   border: 1px solid #ccc;
   padding: 20px;
   h2 {
      font-size: 3em;
      margin: 20px 0;
   }
}
.callout {
   @extend .content;
   background: #ddd;
}
```

main.css

```
.content,
.callout {
   border: 1px solid #ccc;
   padding: 20px;
}
.content h2,
.callout h2 {
   font-size: 3em;
   margin: 20px 0;
}
.callout {
   background: #ddd;
}
```

Extend – problems

main.scss

```
.btn-a {
   background: #777;
   border: 1px solid #ccc;
   font-size: 1em;
   text-transform: uppercase;
}
.btn—b {
   @extend btn-a;
   background: #ff0;
}
.sidebar .btn-a {
   text-transform: lowercase;
}
```

main.css

```
.btn-a,
.btn-b {
   background: #777;
   border: 1px solid #ccc;
   font-size: 1em;
   text-transform: uppercase;
}
.btn—b {
   background: #ff0;
}
.sidebar .btn-a,
.sidebar .btn-b {
   text-transform: lowercase;
}
```

Note that as the .btn-a styles change inside the .sidebar class, the definition of .btn-b within the .sidebar class also changes.

Replacement selector / Placeholder

- As long as .btn-b extends .btn-a class, each instance that modifies .btn-a class also modifies .btn-b class.
- We can use here placeholders
- We declare them using%
- They can be extended, but they are never a selector in and of themselves

Replacement selector / Placeholder

main.scss

```
%btn {
   background: #777;
   border: 1px solid #ccc;
   font-size: 1em;
   text-transform: uppercase;
}
.btn-a {
   @extend %btn;
}
.btn—b {
   @extend %btn;
   background: #ff0;
}
.sidebar .btn-a {
   text-transform: lowercase;
}
```

main.css

```
.btn-a,
.btn-b {
   background: #777;
   border: 1px solid #ccc;
   font-size: 1em;
   text-transform: uppercase;
}
.btn—b {
   background: #ff0;
}
.sidebar .btn-a {
   text-transform: lowercase;
}
```

Functions

main.scss

```
@function fluidize($target, $context) {
   @return ($target / $context) * 100%;
}
.sidebar {
   width: fluidize(350px, 1000px);
}
```

main.css

```
.sidebar {
   width: 35%;
}
```

If, else if, else

main.scss

```
$theme: pink;

header {
    @if $theme == dark {
        background: #000;
    } @else if $theme == pink {
        background: pink;
    } @else {
        background: #fff;
    }
}
```

main.css

```
header {
    background: pink;
}
```

Comparison operators

- == equal
- != not equal
- > greater than
- >= greater than or equal to
- < less than
- <= less than or equal

Loops: each

- With @each we can go through the whole list.

main.scss

```
$authors: maciej pawel michal;

@each $author in $authors {
   .author-#{$author} {
      background: url(author-
#{$author}.jpg)
   }
}
```

main.css

```
.author-maciej {
   background: url(author-
maciej.jpg);
}
.author-pawel {
   background: url(author-
pawel.jpg);
}
.author-michal {
   background: url(author-
michal.jpg);
}
```

Loops: for

main.scss

```
$i: 1;

.item {
   position: absolute;
   right: 0;
   @for $i from 1 through 4 {
      &.item-#{$i} {
         top: $i * 30px;
      }
   }
}
```

main.css

```
.item {
   postion: absoulte;
   right: 0;
}
.item.item-1 {
   top; 30px;
}
.item.item-2 {
   top; 60px;
}
.item.item-3 {
   top; 90px;
}
.item.item-4 {
   top; 120px;
}
```

Loops: while

main.scss

```
$i: 1;

.item {
   position: absolute;
   right: 0;
   @while $i < 4 {
      &.item-#{$i} {
         top: $i * 30px;
      }
      $i: $i + 1;
   }
}
```

main.css

```
.item {
   postion: absoulte;
   right: 0;
}
.item-1 {
   top; 30px;
}
.item-2 {
   top; 60px;
}
.item-3 {
   top; 90px;
}
```

Mathematical Functions

- Using SASS, we can use all numerical operations (they can be applied on every type of data - even colors):
  - Addition +
  - Subtraction -
  - Multiplication *
  - Division /
  - Division modulo %

Mathematical Functions

- `round($number)` – rounding to an integer
- `ceil($number)` – rounding up
- `floor($number)` – rounding down
- `abs($number)` – the absolute value
- `min($list)` – minimum value from the list
- `max($list)` – maximum value from the list
- `percentage($number)` – converting to a percentage

Mathematical Functions

main.scss

```
h2 {
    line-height: ceil(1.2);
}
```

main.css

```
h2 {
    line-height: 2;
}
```

main.scss

```
$context: 1000px;

.sidebar {
    width: percentage(450px/
$context);
}
```

main.css

```
.sidebar {
    width: 45%;
}
```

Thank you for your attention