Sass Functions: A Case Study

Zurb Foundation's emCalc()

Who am I?

- @arachattack on twitter, behance
- @rvinay88
- UI/UX/Sass/CSS
- Coding, Web design, development, Graphic and vector design
- Wordpress, Github, ST2

What we are talking about

- em's in web design
- Foundation – Frontend framework
- Sass – Syntactically awesome style sheets

em's in web design

- em is a unit that is **relative** to the currently chosen **font size**
- used when you specifically want the size of something to depend on the current font size.

Reference<http://stackoverflow.com/questions/609517/why-em-instead-of-px>

Zurb Foundation

- Sass based framework
- Similar to Twitter bootstrap

Sass

- Syntactically awesome style sheets
- Preprocessor
- Binary > Assembly > C/C++ > CSS > Sass

Sass Components

1. Variables
2. Functions
3. Loops and Control Directives
4. Lists
    1. nth function
    2. append function
    3. Length function
5. Arithmetic

Sass components
Variable Declaration

CSS like syntax

```
$a: 10px;
body {
        margin: $a;
}
```

```
body {
  margin: 10px;
}
```

Sass components - Functions

- Takes arguments
- Returns value
- Different from a mixin

```
@function some-calculation($first-number,
$second-number){
  @return $first-number + $second-number
}

.some-div{
  padding: some-calculation(10px, 5px);
}

.some-div {
  padding: 15px;
}
```

Sass components - Functions

- Optional arguments

```
@function some-calculation($first-number,
$second-number: 5px){
  @return $first-number + $second-number
}

.some-div{
  padding: some-calculation(10px);
}

.some-div {
  padding: 15px;
}
```

Sass components - Mixin

- Used to return whole rules instead of just a value

```
@mixin default-padding($some-number) {
  padding: $some-number;
}

.some-div {
  @include default-padding(15px);
}
```

```
.some-div {
  padding: 15px;
}
```

Sass components – Function vs Mixin

- Mixins used to return whole rules instead of just a value

Reference <http://www.developerknowhow.com/getting-to-know-sass/ >

Sass components
Control directives

- @for

```
@for $i from 1 through 3 {
  .item-#{$i} { width: 2em * $i; }
}

.item-1 {
  width: 2em; }
.item-2 {
  width: 4em; }
.item-3 {
  width: 6em; }
```

Sass components
Control directives

- @if

```
$type: monster;
p {
  @if $type == ocean {
    color: blue;
  } @else if $type == matador {
    color: red;
  } @else if $type == monster {
    color: green;
  } @else {
    color: black;
  }
}

p {
  color: green;
```

Sass components
Lists

- Data type in Sass
- Series of values either comma separated or space separated
- Individual values count as lists, too: they're just lists with one item.

```
margin: 10px 20px 30px 40px;
font: Helvetica, Arial, sans-serif;

margin: 10px;
```

List Declaration

```
$emValues: ();
```

List Functions
nth function

```
nth(10px 20px 30px, 1) => 10px
nth((Helvetica, Arial, sans-serif), 3) => sans-serif
```

List Functions
append

```
append(10px 20px, 30px) => 10px 20px 30px
append((blue, red), green) => blue, red, green
```

List Functions
length function

```
length(10px) => 1
length(10px 20px 30px) => 3
```

Sass components
Arithmetic

```
$a: 10px;
$b: $a $a+10 $a+20 $a+30;

body{
    margin: $b;
}
```

```
body {
  margin: 10px 20px 30px 40px; }
```

Sass components
Getting 1 unit

```
$a: 10px;
$b: $a * 0 + 1;

body{
    margin: $b;
}
```

```
body {
  margin: 1px; }
```

Getting Magnitude

```
$a: 10px;
$b: $a / 1px;

body{
    margin: $b;
}
```

```
body {
  margin: 10; }
```

Sass components
Getting 1 unit

```
$a: 10em;
$b: $a * 0 + 1;

body{
    margin: $b;
}
```

```
body {
  margin: 1em; }
```

Getting Magnitude

```
$a: 10em;
$b: $a / ($a * 0 + 1); // 10em / 1em

body{
    margin: $b;
}
```

```
body {
  margin: 10; }
```

Magnitude and Units

```
$a: 10px;
$base: 16px;

body {
  padding: $a / $base;
  margin: ( $a / $base ) * 1em;
}
```

```
body {
  padding: 0.625;
  margin: 0.625em;
}
```

Zurb Foundation's emCalc()

```
$em-base: 16px !default;

@function emCalc($pxWidth) {
  @return $pxWidth / $em-base * 1em;
}

body{
     margin: emcalc(10px);
}
```

```
body {
  margin: 0.625em;
}
```

Limitations of the emCalc()

- Multiple calls for multiple values
- Required the unit, px to be mentioned

```
$em-base: 16px !default;

@function emCalc($pxWidth) {
  @return $pxWidth / $em-base * 1em;
}

body{
     margin: emCalc(10px) emCalc(20px) emCalc(30px) emCalc(40px);
}
```

```
body {
  margin: 0.625em 1.25em 1.875em 2.5em;
}
```

New emCalc()

- Call only once irrespective of number of parameters (1, 2, 3 or 4)
- Need not mention px values
- Need to be backward compatible; should not break when px is mentioned

```
.body{margin: emCalc(10)}
.body{margin: emCalc(10 20)}
.body{margin: emCalc(10 20 30)}
.body{margin: emCalc(10 20 30 40)}

.body{margin: emCalc(10px)}
.body{margin: emCalc(10px 20px)}
.body{margin: emCalc(10px 20px 30px)}
.body{margin: emCalc(10px 20px 30px 40px)}
```

New emCalc()

- Strip Units

```
@function strip-unit($num) {
  @return $num / ($num * 0 + 1);
}

strip-unit(10px) => 10
```

New emCalc()

- Convert to em

```
$em-base: 16 !default;

// Converts "px" to "em" using the ($)em-base

@function convert-to-em($value)  {
  $value: strip-unit($value) / strip-unit($em-base) * 1em;
  @return $value;
}


convert-to-em(16px) => 1em
```

New emCalc()

- emCalc()

```
@function emCalc($values) {
  $max: length($values); // Get the total number of parameters passed

@for $i from 1 through $max {
    $temp: convert-to-em(nth($values, $i));
    $emValues: append($emValues, $temp);
  }
  @return $emValues;
}
```

```
emCalc(16px) => 1em;
emCalc(16 32px) => 1em 2em;
```

New emCalc()

- emCalc()

```
@function emCalc($values) {
  $max: length($values); // Get the total number of parameters passed
  @for $i from 1 through $max {
    $emValues: append($emValues, convert-to-em(nth($values, $i)));
  }
  @return $emValues;
}

emCalc(16px) => 1em;
emCalc(16 32px) => 1em 2em;
```

```
$em-base: 16 !default;


@function strip-unit($num) {
  @return $num / ($num * 0 + 1);
}


@function convert-to-em($value)  {
  $value: strip-unit($value) / strip-unit($em-base) * 1em;
  @return $value;
}


@function emCalc($values) {
  $max: length($values);

  $emValues: ();
  @for $i from 1 through $max {
    $emValues: append($emValues, convert-to-em(nth($values, $i)));
  }
  @return $emValues;
}
```

New emCalc()

- Call only once irrespective of number of parameters (1, 2, 3 or 4)
- Need not mention px values
- Need to be backward compatible; should not break when px is mentioned

```
.body{margin: emCalc(10)}
.body{margin: emCalc(10 20)}
.body{margin: emCalc(10 20 30)}
.body{margin: emCalc(10 20 30 40)}

.body{margin: emCalc(10px)}
.body{margin: emCalc(10px 20px)}
.body{margin: emCalc(10px 20px 30px)}
.body{margin: emCalc(10px 20px 30px 40px)}
```

Issues with the new emCalc()

- 0em instead of 0
- Returning a list is equivalent of returning a string; When there is only one parameter, you can't do emCalc(10px) * 2

```
$em-base: 16 !default;

@function strip-unit($num) {
  @return $num / ($num * 0 + 1);
}

@function convert-to-em($value)  {
  $value: strip-unit($value) / strip-unit($em-base) * 1em;
  @if ($value == 0em) { $value: 0; } // Turn 0em into 0
  @return $value;
}

@function emCalc($values) {
  $max: length($values);

  @if $max == 1 { @return convert-to-em(nth($values, 1)); }

  $emValues: ();
  @for $i from 1 through $max {
    $emValues: append($emValues, convert-to-em(nth($values, $i)));
  }
  @return $emValues;
}
```

Issues with em sizing

- ul li {1.1em}
- ul li li → 2.2em
- Rem font sizing

Handling em Issue

- Sometimes, you need to calculate em on a different base
- Create an optional argument called em-base, which can be passed to the program

```
$em-base: 16 !default;

@function strip-unit($num) {
  @return $num / ($num * 0 + 1);
}

@function convert-to-em($value)  {
  $value: strip-unit($value) / strip-unit($em-base) * 1em;
  @if ($value == 0em) { $value: 0; } // Turn 0em into 0
  @return $value;
}

@function emCalc($values) {
  $max: length($values);

  @if $max == 1 { @return convert-to-em(nth($values, 1)); }

  $emValues: ();
  @for $i from 1 through $max {
    $emValues: append($emValues, convert-to-em(nth($values, $i)));
  }
  @return $emValues;
}
```

```
$em-base: 16 !default;

@function strip-unit($num) {
  @return $num / ($num * 0 + 1);
}

@function convert-to-em($value, $base-value: $em-base)  {
  $value: strip-unit($value) / strip-unit($base-value) * 1em;
  @if ($value == 0em) { $value: 0; } // Turn 0em into 0
  @return $value;
}

@function emCalc($values, $base-value: $em-base) {
  $max: length($values);

  @if $max == 1 { @return convert-to-em(nth($values, 1), $base-value);
}

  $emValues: ();
  @for $i from 1 through $max {
    $emValues: append($emValues, convert-to-em(nth($values, $i), $base-
value));
  }
  @return $emValues;
}
```

Newer emCalc()

- Not merged into foundation yet
- Call with or without optional base em value
- Does not work with commas

```
margin: emCalc(10 20 30 40);
margin: emCalc(10 20 30 40, 16);
margin: emCalc(10 20 30 40, 32);
margin: emCalc(10px 20px 30 40, 32px);

margin: 0.625em 1.25em 1.875em 2.5em;
margin: 0.625em 1.25em 1.875em 2.5em;
margin: 0.3125em 0.625em 0.9375em 1.25em;
margin: 0.3125em 0.625em 0.9375em 1.25em;
```

Questions?

- Let em rip!
- rvinay88@gmail.com

References

http://css-tricks.com/snippets/css/less-mixin-for-rem-font-sizing/

http://net.tutsplus.com/tutorials/other/mastering-sass-lesson-1/

https://developers.google.com/chrome-developer-tools/docs/css-preprocessors