Maintainable CSS

Syntactically Awesome Style Sheets

Preprocessing

Writing a lot of CSS can be overwhelming. Thanks to the CSS pre – processor, it’s now possible to write DRY CSS code.

- Allows you to declare variables that can be re-used all throughout the style sheet.
- Higher level style syntax that provides advanced CSS features.
- Compiled CSS files are uploaded to the production web server.

Sass takes your preprocessed SASS file and save it as a normal CSS file that you can use in your web site. The most direct way to make this happen is in your terminal. You can watch either individual files or entire directories. In addition, you can watch folders or directories with the --watch flag.

```
sass --watch sass_folder:stylesheets_folder
```

sass_folder is the folder of where your sass files are kept (file extensions must be .sass) and stylesheets_folder is your output folder. The --watch option means it’ll watch this folder and if we make any changes to files they will get converted as soon as you save them.

SASS Syntax

- SASS uses indentation instead of { } to delimit code blocks.
- Everything that would be within { and } after a statement must be on a new line and indented one level deeper than that statement.
- Tabs and spaces are not the same even if they look the same!

CSS

```
#main {
  color: blue;
  font-size: 0.3em;
}
```

SASS

```
#main
  color: blue
  font-size: 0.3em
```

http://sass-lang.com/documentation/file.INDENTED_SYNTAX.html

Variables

There are six different types of variables you can use with SASS.

Strings ($myString: “your text here”)
Numbers ($myNum: 10px)
Colors ($myColor: white)
Booleans ($myBool: true)
Lists ($myItemList: 1px solid red)
Nulls ($myVar: null)

Variables

Built-in functions and operators

The paragraphs will be a lighter red than the h1 tags.

```
$red: #FF4848
$fontsize: 12px
h1
  color: $red
p
  color: lighten($red, 10%)
```

Manually darken a color or adjust a font size:

```
  color: $red - #101
  font-size: $fontsize + 10px
```

http://sass-lang.com/documentation/Sass/Script/Functions.html

Nesting

SASS allows you to define nested styles for readability.

CSS

```
#container p {
   font-family: Arial;
   font-size: 13px;
}

#container h1 {
   font-family: Tahoma;
   font-size: 15px;
}

#container h2 {
   font-family: Helvetica;
   font-size: 14px;
}
```

SASS

```
$myFontsize: 13px
$myFontsize2: 15px
$myWidth: 500px
$myMargin: 0px auto

#container
        width: $myWidth
        margin: $myMargin

       p
              font-family: Arial
              font-size: $myFontsize
       h2
                font-family: Helvetica
                font-size: $myFontsize2
```

Mixins

- Mixins let you groups of CSS declarations that can be reused.
- Values an be passed in for more flexibility when working with vendor prefixes.

```
@mixin border-radius($amount: 5px)
  -moz-border-radius: $amount
  -webkit-border-radius: $amount
  border-radius: $amount

h1
  @include border-radius(2px)

.h2
  @include border-radius
```

Property Inheritance

Using inheritance lets you share a set of CSS properties from one selector to another.

SASS

```
.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
```

CSS

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

.success {
  border-color: green;
}

.error {
  border-color: red;
}

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

Q&A

Backup Slides

Variables

Built-in functions

SASS offers a variety of functions. For example:

- darken(color, amount)
- lighten(color, amount)
- saturate(color, amount)
- desaturate(color, amount)
- alpha(color)

http://sass-lang.com/documentation/Sass/Script/Functions.html

Variables

Sass has a handful of standard math operators like +, -, *, /, and %. For example, we can convert px to percentages.

SASS

```
.container
  width: 100%

article
   width: 600px / 960px * 100%

aside
  width: 300px / 960px * 100%
```

CSS

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

article {
  width: 62.5%;
}

aside {
  width: 31.25%;
}
```