Sass :: Pre-compile your CSS

3 August 2015

Links

Enabling Root User On a Mac

http://tutorials.thecodeeducators.com/enabling_root_user_on_mac_os_10.8/

Installing Sass

http://sass-lang.com/install

CSS's Shortcomings

☞ Repeats itself

☞ Separate files must be fetched over the wire

☞ No variables — until now, and they're clumsy

```
:root {
   --primary-bg-color: purple;
}

div {
   color: --primary-bg-color;
   font-size: 1.4rem;
}
```

What is Sass?

- ☞ Syntactically Awesome Stylesheets
- ☞ A superset over CSS3
- ☞ A preprocessor, or precompiler, such as PHP and the one embedded into the C compiler

What is Sass?

☞ A language in two flavors:

① The original syntax is bracket-less, like Python:

```
body
   color: white
```

① The newer syntax, which uses curly braces, is CSS-like:

```
body {
   color: white;
}
```

☞ I'll use the new syntax for this workshop

How Sass Works

☞ Sass preprocesses your CSS and writes a CSS file formatted according to the `style` option:

① `nested`: Reflects the descendant nature of your selector chains

② `compact`: On one line

③ `compressed`: Minified

④ `expanded`: More common. Looks like the CSS you would write.

☞ It watches a source file and writes changes to a target file

☞ The scaffolding of your project remains the same, but the file you edit changes

Compiling

☞ ...Sass is simple:

```
sass  --unix-newlines \
      --sourcemap=none \
      --style expanded \
      --watch source.scss:target.css
```

☞ Consider adding an alias to your shell:

```
alias sass='sass --unix-newlines --sourcemap=none --style expanded --
watch'
```

☞ Now, to compile your Sass, you just write

```
sass source.scss:target.css
```

Compiling

- ☞ You **always** edit your Sass file, **not** your CSS file — Sass writes the latter for you.
- ☞ The command line outputs the status of each update.
- ☞ Errors are output to The Terminal and written to the resulting CSS file via `body:before`.
- ☞ The filename `source.scss` and `target.css` in the previous example are for pedagogical purposes. A more realistic file naming structure might be `style.scss` and `style.css`

File Structure

- ☞ Convention dictates that your CSS be kept in a `css` or `stylesheets` folder, and your Sass files in a `sass` folder.
- ☞ Thus, compiling sass from the root folder of your project might look like:

```
sass sass/style.scss:css/style.css
```

Nesting

Consider the following:

```
<header>
    <nav>
        <ul>
            <li>Bio</li>
            <li>Gallery</li>
        </ul>
    </nav>
</header>
```

Nesting

If you wanted to remove the `list-item-markers`, you'd target the `<ul>` as follows in Sass:

```
header {
    nav {
        ul {
            list-style-type: none;
        }
    }
}
```

Nesting

Resulting in the following CSS...

```
header nav ul {
  list-style-type: none;
}
```

Namespace Shortcuts

```
div {
    background: {
        color: transparent;
        image: url("bg.png");
        repeat: no-repeat;
        attachment: fixed;
        position: top left;
    }
}
```

Namespace Shortcuts

```
div {
  background-color: transparent;
  background-image: url("bg.png");
  background-repeat: no-repeat;
  background-attachment: fixed;
  background-position: top left;
}
```

Namespace Shortcuts

So, this...

```
a {
    :link {
        text-decoration: none;
    }
}
```

is not the same as...

```
a {
    &:link {
        text-decoration: none;
    }
}
```

Namespace Shortcuts

Works with classes, also:

```
p {
    &.highlight {
        background-color: yellow;
    }
}
```

becomes...

```
p.highlight {
  background-color: yellow;
}
```

Combining Nested Elements With Namespace Shortcuts

```
div {
   background: {
      color: transparent;
      image: url("bg.png");
      repeat: no-repeat;
      attachment: fixed;
      position: top left;
   }
   p {
      color: #444;
   }
}
```

Combining Nested Elements With Namespace Shortcuts

Result:

```
div {
  background-color: transparent;
  background-image: url("bg.png");
  background-repeat: no-repeat;
  background-attachment: fixed;
  background-position: top left;
}
div p {
  color: #444;
}
```

☞ Don’t confuse a shortcut with a nested element — the shortcut uses a colon. Omitting the colon from the example in the previous slide means you’re targeting an element `background` that descends from a `div`.

Comments

☞ Normal CSS-style comments:

```
/* Normal CSS-style comments are copied to target file in every
   mode except compressed mode. */
```

☞ Normal CSS-style comments forced through to the compressed copy:

```
/*! Adding an exclamation point as the first character in a
    CSS-style comment forces the comment to be copied to the
    target file in compressed mode. */
```

☞ Single-line comments in Sass are not copied to their target file:

```
// The following CSS is a shame!
```

Variables

```
$color-bg: #bbb;
$color-border: #888;
$color-main: #222;
$font-stack: "Open Sans", "Helvetica Neue", sans-serif;

body {
    background-color: $color-bg;
    border: $color-border;
    color: $color-main;
    font: 1rem $font-stack;
}
```

Darken Colors

```
$color-main: #abcdef;

body {
    color: $color-main;
    border: darken( $color-main, 20% );
}
```

Lighten Colors

```
$color-main: #222;

body {
    color: $color-main;
    border: lighten( $color-main, 32% );
}
```

Mixins

Create...

```
@mixin javascript-code {
    font: 1rem / 1.5 $font-stack-for-code;
    color: blue;
}
```

Use...

```
code.js-example {
    @include javascript-code;
}
```

Mixins as Functions

```
@mixin transition( $property, $duration, $timing_function, $delay ) {
   -webkit-transition: $property $duration $timing_function $delay;
      -moz-transition: $property $duration $timing_function $delay;
       -ms-transition: $property $duration $timing_function $delay;
        -o-transition: $property $duration $timing_function $delay;
           transition: $property $duration $timing_function $delay;
}

p {
   color: red;
}

p:hover {
   @include transition( color, 1s, ease-in, 500ms );
}
```

Mixins as Functions with Default Arguments

```
@mixin transition( $prop: all, $dur: 1s, $time_func: ease, $del: 0s ) {
   -webkit-transition: $prop $dur $time_func $del;
      -moz-transition: $prop $dur $time_func $del;
       -ms-transition: $prop $dur $time_func $del;
        -o-transition: $prop $dur $time_func $del;
           transition: $prop $dur $time_func $del;
}


p {
   color: red;
}

p:hover {
   @include transition( $prop: color );
}
```

Media Queries

```
body {
   width: 90%;

   @media screen and (max-width: 960px) {
      width: 95%;
   }
}
```

Becomes...

```
body {
  width: 90%;
}
@media screen and (max-width: 960px) {
  body {
    width: 95%;
  }
}
```

Media Queries with Variables

```
$width-phone: 320px;
$width-laptop: 1366px;

body {
   width: 90%;

   @media screen and (max-width: $width-laptop) {
      width: 95%;
   }

   @media screen and (max-width: $width-phone) {
      width: 100%;
   }
}

main {
   width: 100%;

   @media screen and (max-width: $width-laptop) {
      padding: 32px;
   }

   @media screen and (max-width: $width-phone) {
      padding: 4px;
   }
}
```

Media Queries with Variables

```
$monochrome-printer: "print and (monochrome)";

body {
    font-size: 1.4rem;

    @media #{$monochrome-printer} {
        font-size: 12pt;
    }
}

main {
    color: #222;

    @media #{$monochrome-printer} {
        color: black;
    }
}
```

Media Queries with Variables

Sass input:

```
$large-screen: 1024px;
$medium-screen: 800px;
$small-screen: 320px;

@mixin responsive( $width ) {
   @if $width == large {
      @media only screen and ( max-width: $large-screen ) {
         @content;
      }
   }
   @else if $width == medium {
      @media only screen and ( max-width: $medium-screen ) {
         @content;
      }
   }
   @else if $width == small {
      @media only screen and ( max-width: $small-screen ) {
         @content;
      }
   }
}
```

Media Queries with Variables

```
main {
    @include responsive( large ) {
        color: red;
        width: 200px;
    }
    @include responsive( medium ) {
        border: 1px solid lighten( red, 20% );
        width: 200px;
    }
    @include responsive( small ) {
        color: blue;
        width: 200px;
    }
}
```

Media Queries with Variables

CSS Output:

```
main {
  color: #222;
}
@media print and (monochrome) {
  main {
    color: black;
  }
}

@media only screen and (max-width: 1024px) {
  main {
    color: red;
    width: 200px;
  }
}
@media only screen and (max-width: 800px) {
  main {
    border: 1px solid #ff6666;
    width: 200px;
  }
}
@media only screen and (max-width: 320px) {
  main {
    color: blue;
    width: 200px;
  }
}
```

External Files

- ☞ Use `@import "FILE.scss"` to combine the contents of `FILE.scss` with the current file
- ☞ Doing so decreases HTTP requests
- ☞ Use `@import url("FILE.css")` as normal to import multiple CSS files
- ☞ A good Sass external file structure is...

① Import a base/reset/normalize style sheet: `_base.scss`

② Import mixins: `_mixins.scss`

③ Import variables: `_variables.scss`

External Files

```
@import "_base.scss";
@import "_mixins.scss";
@import "_variables.scss";
/* ------------------------------------------------------------ *\
   Perform sassy magic here…
\* ------------------------------------------------------------ */
```

Extending Rules

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

.urgent {
    @extend .warning;
    color: red;
}
```

① Add `.urgent` to the element list containing `.warning`

② Set the foreground color of red only to the `.urgent` class

Extending Rules

Becomes...

```
.warning, .urgent {
  background-color: yellow;
}

.urgent {
  color: red;
}
```

Extending Rules

Does not work like...

```
.warning.urgent {
    background-color: yellow;
}
```

Miscellaneous

☞ Sass will check for some code issues, such as unbalanced curly brances, but it won't check some others:

```
french {
    &:fries {
        are: delicious;
    }
}
```

Miscellaneous

Becomes…

```
french:fries {
  are: delicious;
}
```

Miscellaneous

☞ Empty rules in Sass are not copied to the target CSS file

Helpful Resources

☞ Chris Coyier's Sass Style Guide

☞ Dan Cederholm's Sass for Web Designers