Less CSS

Objectives

- Install and use Less to generate CSS
- Increase maintainability by using Less language features

What is Less?

- CSS pre-processor
  - input is written in the Less language syntax
  - output is in the CSS language syntax
- Less extends CSS' syntax with useful language features
  - variables
  - math
  - functions
  - nested rules
  - mixins
  - guards
  - extends

Compiling Less

- Reference compiler is implemented with Node.js and distributed via npm
  - comes with command-line compiler: lessc
  - can run in browser
  - many ports to other platforms exist
- Server-side frameworks can compile Less and cache CSS on first request
  - .less for ASP.NET
  - less.js middleware for Connect/Express
- IDEs can compile on save
  - Visual Studio with Web Essentials plugin
  - IntelliJ/WebStorm with LESS CSS Compiler plugin
- Developer tools can watch files and compile on change

Getting Started

- Install Node.js
- Install less via npm

```
npm install -g less
```

- Create .less file
- Compile with lessc

```
lessc style.less > style.css
```

- Include .css file in browser

Variables

- Declare variables with @ prefix

```
@primary-color: #428bca;

main {
    background-color: @primary-color;
}
```

- Variables declared in nested scopes override variables in outer scopes
  - last declaration in scope overrides all previous declarations
- Variables are untyped, values are typed
  - number, string, color, keyword, url, percentage
  - numbers can have units

Math

- Expressions evaluated during compilation

```
@base: 5%;
@filler: @base * 2;
@other: @base + @filler;

color: #888 / 4;
background-color: @base-color + #111;
height: 100% / 2 + @filler;
```

Functions

- Built-in function library for manipulating values

```
@base: #f04615;
@width: 0.5;

.class {
    width: percentage(0.5);
    color: saturate(@base, 5%);
    background-color: spin(lighten(@base,
25%), 8);
}
```


Built-in Functions

- Math
  - percentage(), round(), ceil(), floor(), min(), max(), mod(), pow(), sqrt(), sin(), cos(),
- Color channel
  - red(), green(), blue(), alpha(), hue(), saturation(), lightness(), etc
- Color operation
  - saturate(), desaturate(), lighten(), darken(), fade(), mix(), greyscale(), etc
- Color blending
  - multiply(), screen(), overlay(), softlight(), hardlight(), difference(), etc
- String
  - escape(), e(), %()
- List
  - length(), extract()
- Type
  - isnumber(), isstring(), iscolor(), etc

Nested Rules

- Less prefixes nested rules with outer selectors
  - much easier to maintain

```
#header {
    color: black;

    .navigation {
        font-size: 12px;
    }

    .logo {
        width: 300px;
    }
}
```

Parent Selector

- & references parent selector
  - use with pseudo-classes and elements

```
blockquote {
    border-left: 10px solid #ccc;
    padding-left: 10px;

    &:before {
        content: "\201C"; /* left double
quotation mark */
        font-size: 4em;
        color: #ccc;
    }
}
```

Media Bubbling

- @media at-rules can be nested
  - @media is removed and outer selector is nested inside

```
#logo {
    width: 90%;

    @media (min-width: 768px) {
        float: left;
        width: percentage(1/3);
    }
}
```

```
#logo { width: 90%; }

@media (min-width: 768px) {
    #logo {
        float: left;
        width: percentage(1/3);
    }
}
```

Mixins

- Re-use common rules

```
.bordered {
    border-top: dotted 1px black;
    border-bottom: solid 2px black;
}

#menu a {
    color: #111;
    .bordered;
}

.post a {
    color: red;
    .bordered;
}
```

Mixin Parameters

- **Mixins parameters look like functions**
  - use semicolons are commas to separate multiple parameters
  - parameters can have default values
  - parameters can be supplied by position or by name

```
.border-radius(@radius: 5px) {
  -webkit-border-radius: @radius;
     -moz-border-radius: @radius;
          border-radius: @radius;
}

#header {  .border-radius; }

.button {  .border-radius(6px);}
```

Mixin Guards

- Feels like method overloading
  - expressions evaluated at compile-time

```
.mixin (@a) when (lightness(@a) >= 50%) {
  background-color: black;
}
.mixin (@a) when (lightness(@a) < 50%) {
  background-color: white;
}
.mixin (@a) {
  color: @a;
}
```


CSS Guards

- Compile-time “if” statements around selectors
  - options be defined as normal variables
  - or with --global-var and --modify-var command-line options

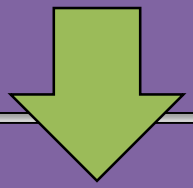
```
button when (@my-option = true) {
  color: white;
}
```

:extend() pseudo-class

- **Alternative to mixins**
  - copies selector, not properties

```
.inline {
  color: red;
}

nav ul {
  &:extend(.inline);
  background: blue;
}
```



```
.inline,
nav ul {
  color: red;
}

nav ul {
  background: blue;
}
```

Summary

- Less extends CSS syntax with widely desired features
  - makes maintaining large CSS code bases much more manageable