CSS3 and Sass

Mark Volkmann
Object Computing, Inc.
mark@ociweb.com
Strange Loop 2011

OBJECT COMPUTING, INC.

CSS

- Separates formatting from content

- Can specify formatting for

  - all elements with a given name

  - all elements with a given class

  - the element with a given id

  - elements with certain relationships

- Major topics to learn

  - selectors

  - properties

  - layout

- W3C documents

  - CSS1 Recommendation (1996)

  - CSS2 Recommendation (1998)

  - CSS3 - defined by many documents at various stages of approval; many are Working Drafts

Three Ways To Specify

- 1) HTML element **style** attribute

  - ex. `<div style="color: red; font-size: 18pt;">Some Text</div>`

  - discouraged

- 2) **style** element

  - inside **head** element in HTML

  - ex. `<style type="text/css"> ...rules... </style>`

  - discouraged

- 3) **link** tag

  - inside **head** element in HTML

  - references an external .css file containing rules

  - ex. `<link rel="stylesheet" type="text/css" href="filename.css"/>`

  - recommended

- Evaluated in the order listed above

  - first property value found for each element wins; important when there are conflicts

Rules

- Specify properties to be applied to selected elements

- More than one rule can be applied to the same element

- Format

  - *selector-list { property-list }*

  - *selector-list* is a comma-separated list of selectors

  - *property-list* is a list of name/value pairs

    - names and values are separated by a colon

    - pairs are separated by a semi-colon

    - while not required, convention is to include a semi-colon after last pair

- Example

  ```
  .header, .title {
    color:     purple;
    font-size: 24pt;
  }
  ```

- Comments

  ```
  /* comment-text */
  ```

Selectors ...

- By element name

  - just the name - for example, `div`, `p`, `td`

- By class name

  - matches elements that have specified value for their `class` attribute

    - for example, `<div class="title">` applies to all div tags that have a class of "title"

  - class name preceded by a period in definition - for example, `.title`, `.header`

  - can also specify element name - for example, `div.title`

- By id

  - matches the element that has specified value for its `id` attribute

    values of id attributes must be unique within the document regardless of the element name

    - for example, `<div id="education">`

  - id preceded by a pound sign - for example, `#date`, `#tax`

  - can also specify element name - for example, `div#date`

... Selectors

- By context - descendants
  - separate with spaces - for example, `div.fy2010 table.taxes td`

- By context - children
  - `a > b` selects elements named "b" that are direct children of an element named "a"
  - for example, `div.fy2010 > table > td` `td` not inside a `thead`, `tfoot` or `tbody`

- There are more, but that's enough for an overview

Box Model

- Many properties are related to the box model of elements

- Margin - space between border and neighboring elements

- Border - can specify type, thickness and color

  - types: `dashed, dotted, double, groove, inset, outset, ridge` or `solid`

  - thickness is specified with a length, typically in pixels (px)

- Padding - space between content and border

margin

padding

content

← border

demo.html ...

```html
<html>
  <head>
    <title>CSS Demo</title>
    <link rel="stylesheet" type="text/css" href="demo.css"/>
  </head>
  <body>
    <div class="title">CSS Demo</div>
    <div class="whippet" id="first">
      <span class="name">Rudy</span>
      <img src="rudy.jpg" alt="Rudy"
        height="100"/>
    </div>
    <div class="whippet" id="second">
      <span class="name">Dasher</span>
      <img src="dasher.jpg" alt="Dasher"
        height="100"/>
    </div>
```


CSS Demo

Rudy

Dasher

I wonder if I've been changed in the night? Let me think. Was I the same when I got up this morning? I almost think I can remember feeling a little different. But if I'm not the same, the next question is 'Who in the world am I?' Ah, that's the great puzzle!

Red Green Blue
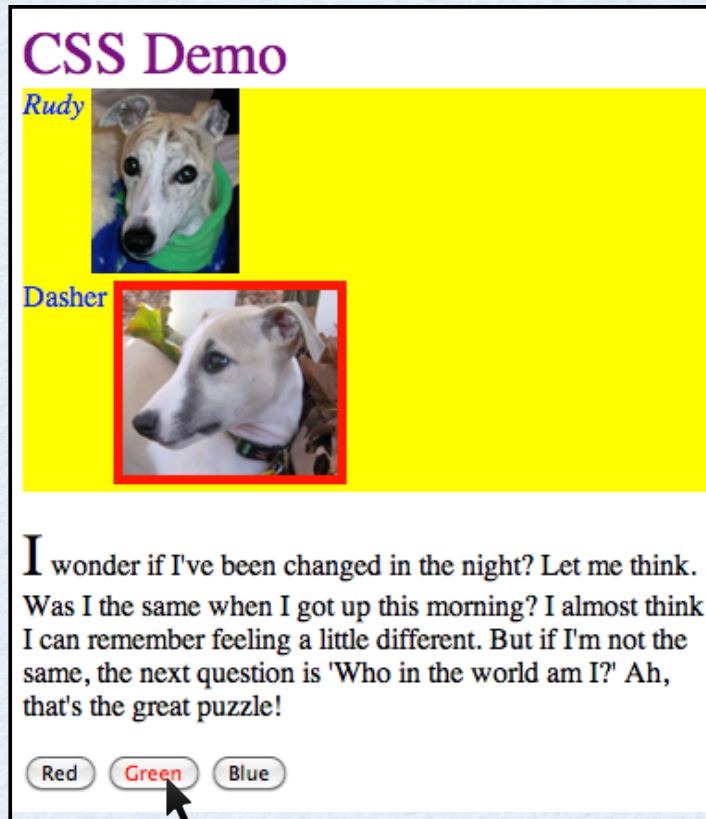
... demo.html

```
<p class="quote">
  I wonder if I've been changed in the night? Let me think.
  Was I the same when I got up this morning?
  I almost think I can remember feeling a little different.
  But if I'm not the same, the next question is 'Who in the world am I?'
  Ah, that's the great puzzle!
</p>
<form method="post" action="">
  <div>
    <button>Red</button>
    <button>Green</button>
    <button>Blue</button>
  </div>
</form>
</body>
</html>
```


CSS Demo

Rudy

Dasher

I wonder if I've been changed in the night? Let me think. Was I the same when I got up this morning? I almost think I can remember feeling a little different. But if I'm not the same, the next question is 'Who in the world am I?' Ah, that's the great puzzle!

Red Green Blue

demo.css ...

```css
.quote {
  width: 370px;
}

.quote:first-letter {
  font-size: 24pt;
}

.title {
  color: purple;
  font-size: 24pt;
}

.whippet {
  background-color: yellow;
  color: blue;
}

.whippet span {
  vertical-align: top;
}
```

a pseudo-class

CSS Demo

Rudy

Dasher

I wonder if I've been changed in the night? Let me think. Was I the same when I got up this morning? I almost think I can remember feeling a little different. But if I'm not the same, the next question is 'Who in the world am I?' Ah, that's the great puzzle!

Red Green Blue

... demo.css

```
#first {
    font-style: italic;
}

#second img {
    border: solid red 5px;
}

button:hover {
    color: red;
}
```

a pseudo-class

CSS Demo

Rudy

Dasher

I wonder if I've been changed in the night? Let me think. Was I the same when I got up this morning? I almost think I can remember feeling a little different. But if I'm not the same, the next question is 'Who in the world am I?' Ah, that's the great puzzle!

Red   Green   Blue

CSS3

- Defined by a long list of modularized W3C specifications
  - at various stages of specification
  - at various stages of browser implementation

- Current browsers
  - don't support some CSS3 features
  - treat others as experimental
  - IE support is very limited
  - Firefox, Chrome and Safari support is better
    -

CSS3 Experimental?

- Rather than support new CSS property names (and some property values), browsers support alternate names that begin with

  - `-ms-` for IE

  - `-moz-` for Firefox

  - `-o-` for Opera

  - `-webkit-` for Safari, iPhone, iOS, Chrome, Android

- When features are no longer considered experimental, unprefixed names will be supported

- Example

  - for now instead of `box-shadow`,
    use `-webkit-box-shadow` or `-moz-box-shadow`

CSS3 Documents Past WD

- **Recommendations**

  from http://www.w3.org/standards/techs/css on 9/4/11

  - A MathML for CSS Profile - 07 June 2011

  - CSS Color Level 3 - 07 June 2011

- **Proposed Recommendations**

  - CSS Namespaces - 11 August 2011

  - Selectors Level 3 - 15 December 2009

- **Candidate Recommendations**

  - CSS Multi-column Layout - 12 April 2011

  - CSS Backgrounds and Borders Level 3 - 15 February 2011

  - CSS Style Attributes - 12 October 2010

  - Media Queries - 27 July 2010

  - Selectors API Level 1 - 22 December 2009

  - CSS Mobile Profile 2.0 - 10 December 2008

  - CSS Marquee Level 3 - 5 December 2008

  - CSS3 Basic User Interface - 11 May 2004

  - CSS TV Profile 1.0 - 14 May 2003

CSS3 Working Drafts

CSS Speech - 18 August 2011

CSS Print Profile - 13 October 2006

CSS3 Paged Media - 10 October 2006

CSS Text Level 3 - 1 September 2011

CSS Writing Modes Level 3 - 1 September 2011

CSS Conditional Rules Level 3 - 1 September 2011

CSSOM View - 4 August 2011

CSS Image Values and Replaced Content Level 3 - 12 July 2011

CSSOM (CSS Object Model) - 12 July 2011

CSS3 Ruby - 30 June 2011

CSS Regions - 09 June 2011

CSS Lists and Counters Level 3 - 24 May 2011

CSS Fonts Level 3 - 24 March 2011

Flexible Box Layout - 22 March 2011

CSS Generated Content for Paged Media - 08 June 2010

CSS Template Layout - 29 April 2010

Selectors API Level 2 - 19 January 2010

CSS 2D Transforms Level 3 - 01 December 2009

CSS Transitions Level 3 - 01 December 2009

CSS 3D Transforms Level 3 - 20 March 2009

CSS Animations Level 3 - 20 March 2009

Behavioral Extensions to CSS - 19 October 2007

CSS Grid Positioning Level 3 - 5 September 2007

CSS Basic Box Model - 9 August 2007

CSS3 Values and Units - 19 September 2006

CSS3 Cascading and Inheritance - 15 December 2005

CSS3 Hyperlink Presentation - 24 February 2004

The CSS 'Reader' Media Type - 24 February 2004

CSS3 Presentation Levels - 13 August 2003

CSS3 Syntax - 13 August 2003

CSS3 Generated and Replaced Content - 14 May 2003

CSS3 Border - 7 November 2002
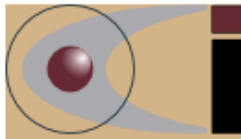
CSS3 Line - 15 May 2002

Media Queries

- W3C Candidate Recommendation 27 July 2010

  - http://www.w3.org/TR/css3-mediaqueries/

  - "media-dependent style sheets tailored for different media types ...
    extend the functionality of media types by allowing more precise labeling of style sheets"

  - "A media query consists of a media type and zero or more expressions
    that check for the conditions of particular media features."

Fonts Level 3

- W3C Working Draft 24 March 2011

  - http://www.w3.org/TR/2011/WD-css3-fonts-20110324/

  - "describes how font properties are specified and how font resources are loaded dynamically"

@ font-face Demo
by R. Mark Volkmann

Text Level 3

- W3C Working Draft 01 September 2011

  - http://www.w3.org/TR/css3-color/

  - "defines properties for text manipulation and specifies their processing model ... covers line breaking, justification and alignment, white space handling, text decoration and text transformation."

- Adds many new properties

  - but most aren't well supported yet

  - only `text-shadow` and a WebKit variant of `text-stroke` are covered

Multi-column Layout

- W3C Candidate Recommendation 12 April 2011
  - http://www.w3.org/TR/css3-multicol-20110412/
  - "content can be flowed into multiple columns with a gap and a rule between them"

Beverly Hillbillies

Come and listen to a story about a man named Jed A poor mountaineer, barely kept his family fed, Then one day he was shootin at some food, And up through the ground came a bubblin crude. Oil that is, black gold, Texas tea.

Well the first thing you know ol Jed's a millionaire, Kinfolk said "Jed move away from there" Said "Californy is the place you ought to be" So they loaded up the truck and moved to Beverly. Hills, that is. Swimmin pools, movie stars.

Well now its time to say goodbye to Jed and all his kin. And they would like to thank you folks fer kindly droppin in. You're all invited back a gain to this locality To have a heapin helpin of their hospitality Hillybilly that is. Set a spell, Take your shoes off.

Y'all come back now, y'hear?.

Backgrounds and Borders

- W3C Candidate Recommendation 15 February 2011

  - http://www.w3.org/TR/css3-background/

  - "includes and extends the functionality of CSS level 2 ...
    The main extensions are borders consisting of images, boxes with multiple backgrounds,
    boxes with rounded corners and boxes with shadows."

Beach Haiku

Balmy wind drifts by
Evening sun kissing the sea
Footprints in the sand

Life is good when you dance all night
And the world transmits electric power
Messages you send to Mars
Came from a crown of flowers

Laura Veirs - "Life Is Good Blues"

The Web site you seek
Cannot be located, but
Countless more exist.

Out of memory.
We wish to hold the whole sky,
But we never will.

Image Values and Replaced Content Level 3

- W3C Working Draft 12 July 2011

  - http://www.w3.org/TR/2011/WD-css3-images-20110712/

  - "two parts …
    defines the syntax for image values in CSS …
    defines properties used to control the interaction of replaced content and the CSS layout algorithms"

  - gradients are a subset of the first part
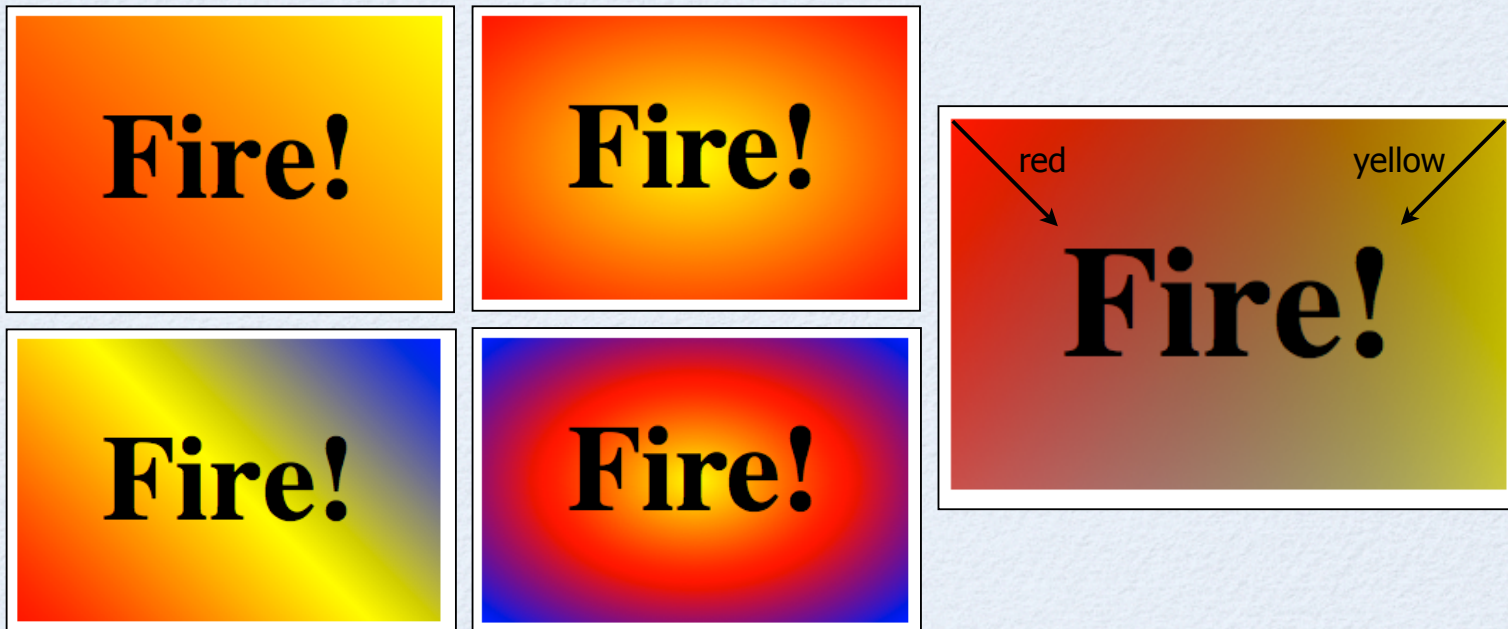
Flexible Box Layout ...

- W3C Working Draft 22 March 2011

  - http://www.w3.org/TR/2011/WD-css3-flexbox-20110322/

  - "In this new box model, the children of a box are laid out either horizontally or vertically,
    and unused space can be assigned to a particular child or distributed among the children
    by assignment of "flex" to the children that should expand.
    Nesting of these boxes (horizontal inside vertical, or vertical inside horizontal)
    can be used to build layouts in two dimensions.
    This model is based on the box model in the XUL user-interface language
    used for the user interface of many Mozilla-based applications (such as Firefox)."

- Also referred to as "Flex Box"

- Greatly simplifies layout

  - over using properties like float and position

FlexBox Centering

Centered Content

E3 E1 E2

E5 E4

E6
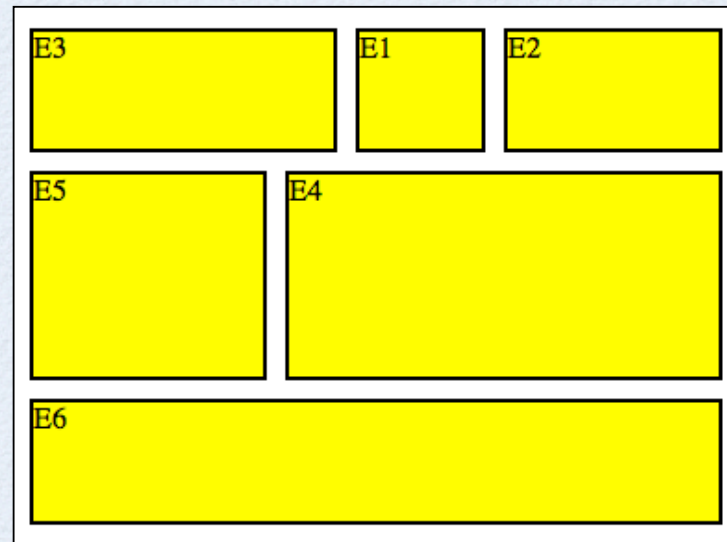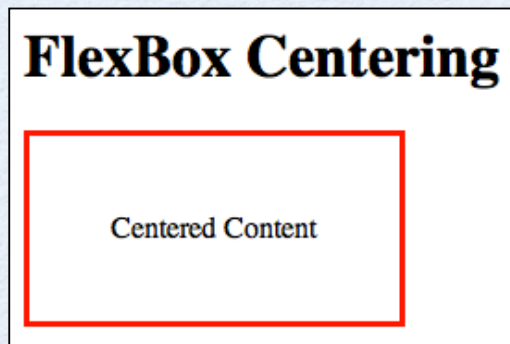
… Flexible Box Layout

Alice's Adventures in Wonderland by Lewis Carroll

CHAPTER I. Down the Rabbit-Hole

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book,' thought Alice 'without pictures or conversation?'

So she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.

See this book on Project Gutenberg.

Transitions Level 3

- W3C Working Draft 01 December 2009

  - http://www.w3.org/TR/2009/WD-css3-transitions-20091201/

  - "allows property changes in CSS values to occur smoothly over a specified duration"

    - rather than immediately and all at once

- Can transition any "animatable CSS property"

2D Transforms

- W3C Working Draft 01 December 2009
    - http://www.w3.org/TR/2009/WD-css3-**2d**-transforms-20091201/
    - "allows elements rendered by CSS to be transformed in two-dimensional space"

3D Transforms

- W3C Working Draft 20 March 2009

  - http://www.w3.org/TR/2009/WD-css3-**3d**-transforms-20090320/

  - "extends CSS Transforms to allow elements rendered by CSS to be transformed in three-dimensional space"
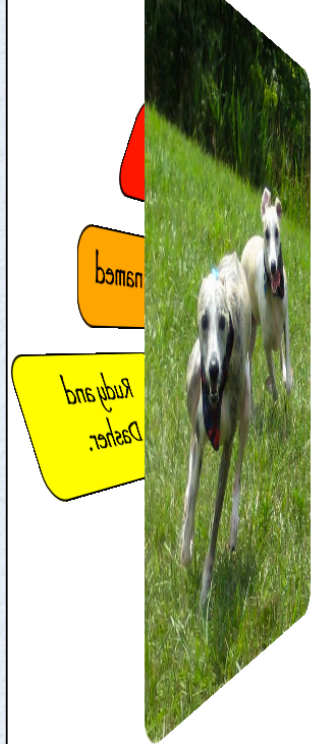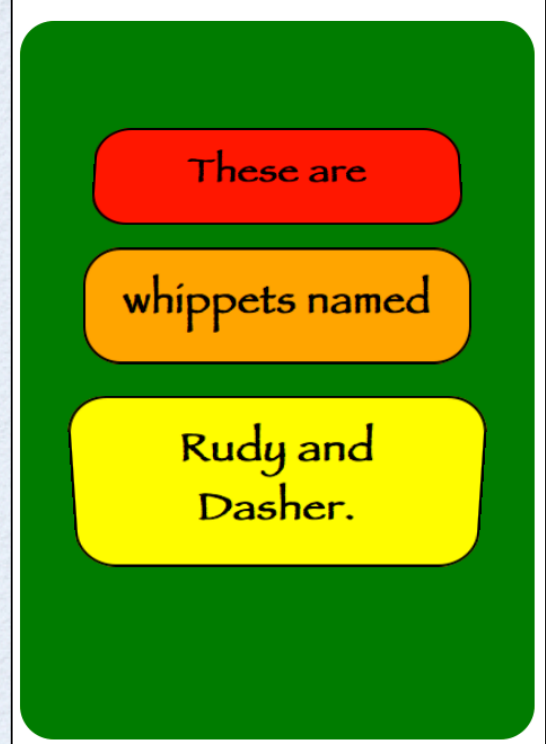
Sass

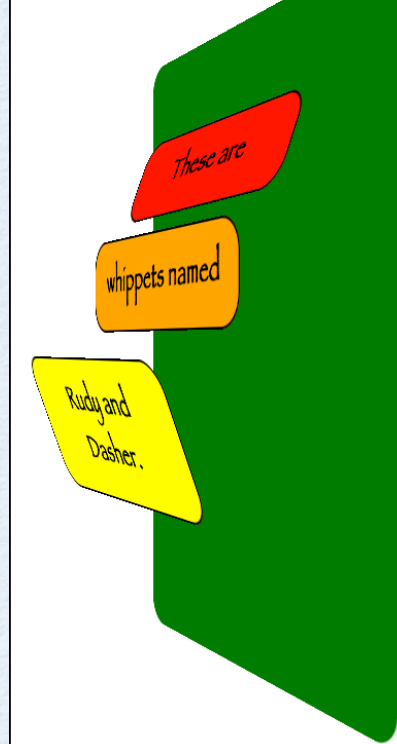- Stands for "Syntactically Awesome StyleSheets"

- Converts an alternate CSS syntax to the W3C CSS syntax

  - two alternate syntaxes

  - a Ruby application, but understanding Ruby isn't required

- Main features

  - **variables** - avoids repeating property values; supports math

  - **nesting** - avoids repeating selectors

  - **mixins** - reuses lines of CSS; optionally parameterized

  - **selector inheritance** - a selector can inherit styles from another

- Created by

  - Hampton Catlin, Nathan Weizenbaum, Chris Eppstein and others

- http://sass-lang.com

Installing Sass

- Install Ruby if not already installed
  - can check with "`ruby -v`"
  - see http://www.ruby-lang.org/en/downloads/
  - need version 1.8.7 or above

- Run "`gem install sass`"
  - example output

  ```
  Successfully installed sass-3.1.3

  1 gem installed

  Installing ri documentation for sass-3.1.5...

  Installing RDoc documentation for sass-3.1.5...
  ```

- Alternative ... from Git

  ```
  git clone git://github.com/nex3/sass.git

  cd sass

  rake install
  ```

Running Sass

- Get version
  - `sass -v -> Sass 3.1.5 (Brainy Betty)`

- Output to stdout
  - `sass {name}.scss`
  - useful for testing

  > by default, creates a `.sass-cache` directory that stores compiled templates to speed up subsequent runs

- Output to a CSS file
  - `sass {name}.scss:{name}.css`

- Other options described later

- Errors in input file
  - described at command-line and in generated `.css` files

- Link to generated `.css` files in `.html` files, not `.scss` files
  - `<link rel="stylesheet" type="text/css" href="{name}.css"/>`

Two Syntaxes

- Sass syntax
  - the original
  - more concise
    - uses indentation instead of braces and newlines instead of semicolons
  - file extension is `.sass`

- SCSS syntax
  - "Sassy CSS"
  - extension of CSS3 syntax
    - all CSS3 syntax is valid SCSS syntax
  - file extension is `.scss`
  - preferred

- Syntax used is indicated to `sass` command by the file extension

Syntax Comparison

```scss
#main {                          SCSS
  color: blue;

  font-size: 18pt;

  a {

    font: {

      weight: bold;

      family: serif;

    }

    &:hover {

      background-color: #eee;

    }

  }

}
```

```sass
#main                            Sass
  color: blue

  font-size: 18pt

  a

    font:

      weight: bold

      family: serif

    &:hover

      background-color: #eee
```

```css
#main {                  generated CSS

  color: blue;

  font-size: 18pt; }

#main a {

  font-weight: bold;

  font-family: serif; }

#main a:hover {

  background-color: #eee; }
```

illustrates use of **nested rules**
which are described later;
& inserts the **parent selector** (**a** in this case)
in front of a pseudoclass (**:hover** in this case)

Syntax Conversion

- From Sass to SCSS

  - `sass-convert {name}.sass {name}.scss`

- From SCSS to Sass

  - `sass-convert {name}.scss {name}.sass`

`style` Option

- Controls style of CSS output

- Values are

  - **nested** - logically nested; default

  - **expanded** - closest to hand-written CSS

    - my preference

  - **compact** - each rule on one line

  - **compressed** - everything on one line

    - comments removed

- To set from command-line

  - **--style={*value*}**

  - **-t={*value*}**

```
#main {                          nested
  color: #fff;
  background-color: #000; }
  #main p {
    width: 10em; }


.huge {
  font-size: 10em;
  font-weight: bold;
  text-decoration: underline; }
```

```
#main {                        expanded
  color: #fff;
  background-color: #000;

}
#main p {
  width: 10em;

}


.huge {
  font-size: 10em;
  font-weight: bold;
  text-decoration: underline;

}
```

`watch` Option

- Watches input files for changes and automatically generates new `.css` files

  - don't have to manually run `sass` command after each change

  - just perform a browser refresh to test changes

- To watch a specific file for changes

  - `sass --watch {name}.scss:{name}.css`

- To watch a directory for changes

  - `sass --watch {sass-dir}:{css-dir}`

  - example: `sass --watch stylesheets/sass:stylesheets/css`

  - if Sass files are in current directory and
    generated `.css` files should be placed there also,
    `sass --watch .:.`

Comments

- Like CSS, Sass supports multiline comments

  - */* comment */*

  - copied to generated .css files

- Sass also supports single line comments

  - *// comment*

  - not copied to generated `.css` files

Nesting

- Useful when multiple selectors specify the same ancestor(s)

  - avoids repeating ancestors in each selector

```
#foo td {                    CSS
  border: solid red 2px;
}
#foo li {
  font-style: italic;

}
```

```
#foo {                      SCSS
  td {
    border: solid red 2px;
  }
  li {
    font-style: italic;

  }

}
```

- Alternative to shorthand properties

```
#foo {                           CSS
  font-family: Tahoma, sans-serif;
  font-size: 18pt;
  font-style: italic;
  font-weight: bold;

}
```

```
#foo {                     CSS shorthand
  font: italic bold 18pt Tahoma, sans-serif;

}
```

```
#foo {                          SCSS
  font: {
    family: Tahoma, sans-serif;
    size: 18pt;
    style: italic;
    weight: bold;

  }

}
```

see border properties example at
http://sass-lang.com/tutorial.html

SassScript

- Extensions to CSS that support

  - variables

  - arithmetic operators: + – * / %

  - computed property values

  - interpolation for computed selectors, property names and property values

  - additional functions beyond those in CSS

  - directives

    - control: `@if`, `@else if`, `@else`, `@for`, `@each`, `@while`

    - mixin: `@mixin`, `@include`

    - other: `@debug`, `@warn`, `@function`

Variables

- Avoid repeating values and
make changes across multiple usages easier

    - ex. instead of changing `red` to `green` in many places, change value of a variable in one place

- Names begin with `$`

- Assigned like properties

    - `$name: value;`

    - to assign a value only if it doesn't already have one, `$name: value !default;`

- Values can include mathematical operators and calls to functions

    - operators are `+ - * / %` with normal precedence

    - parentheses can be used to control evaluation order

    - functions are covered later

- Scoped to CSS rule in which defined

    - global if not defined inside any rule

Variables Example ...

Sass Variables
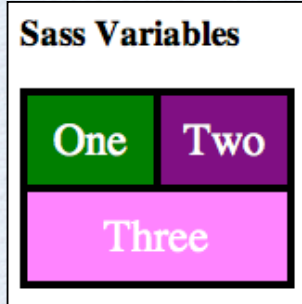
One	Two
Three	

```
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="variables.css"/>
  </head>
  <body>
    <h3>Sass Variables</h3>
    <div id="d1">One</div>
    <div id="d2">Two</div>
    <div id="d3">Three</div>
  </body>
</html>
```

... Variables Example

```scss
$borderWidth: 4px;
$color1: green;
$padding: 10px;
$width: 120;
$halfWidth: $width/2 - $padding - $borderWidth/2;

div {
  border: solid black $borderWidth;
  color: white;
  font: 18pt bold monospace;
  padding: $padding;
  text-align: center;
}
```

Sass Variables

One	Two
Three	

```scss
#d1 {
  background-color: $color1;
  float: left;
  width: $halfWidth;
}
#d2 {
  background-color: complement($color1);
  border-left-width: 0;
  float: left;
  width: $halfWidth;
}
#d3 {
  background-color: invert($color1);
  border-top-width: 0;
  clear: left;
  width: $width;
}
```

Interpolation

- Used to compute
  - selectors
  - property names
  - property values

- Syntax is **#{*expression*}**

- Example

```scss
$planet: Mars;                                    SCSS
$number: 3;
$side: top;


.#{$planet}#{$number + 1} {
  border-#{$side}-width: #{$number * 2}pt;
}
```

```css
.Mars4 {   generated CSS
  border-top-width: 6pt;
}
```

Data Types

- Boolean
  - literal values **true** and **false** and operators **and, or** and **not**

- Number
  - can have CSS units like **pt, px** and **in** - ex. **24pt**
  - arithmetic operations perform automatic conversions `2in - 4mm = 1.843in`

- String
  - in double-quotes, single-quotes or unquoted; use + to concatenate

- Color
  - name, hex value or call to one of many functions that return a color

- List
  - values separated by spaces or commas
  - some properties take a list value
    - examples include **margin, padding** and **border-color**
      (order is top, right, bottom, left which is clockwise from 12 o'clock)
    - **border-color: red orange yellow green;**

Functions

- Strings
  - **quote, unquote**
    - adds/removes outer quotes if they don't/do exist
- Math
  - **abs, ceil, floor, round, percentage** (simply multiplies by 100)
- Lists
  - **join, length, nth**
- Introspection
  - **comparable** (takes two numbers with optional units and returns true if they can be compared, added and subtracted)
  - **type-of, unit, unitless**

  > for example, an absolute length cannot be compared to a relative one; **comparable(50px, 5ex)** returns false

- Other
  - **if(*condition*, *true-value*, *false-value*)**
    - not very useful in practice because Sass doesn't have access to dynamic information such as time, date, user preferences, ...

Color-related Functions

returns the amount of these (0 to 255) in a given color

- RBG
  - **rgb**, **rgba**, **red**, **green**, **blue**, **mix**

    mix(color1, color2, weight:50%) weight is percentage of color1

- HSL
  - **hsl**, **hsla**, **hue**, **saturation**, **lightness**
  - **adjust-hue**, **complement**, **grayscale**, **invert**, **lighten**, **darken**
  - **saturate**, **desaturate**

- Opacity
  - **alpha** (or **opacity**), **rgba**, **opacify** (or **fade-in**), **transparentize** (or **fade-out**)

- Other color-related
  - **adjust-color**, **scale-color**, **change-color**

    changes one or more properties of a color; see example on next slide

Function Calls

- Can call with positional or keyword arguments

- Keyword arguments are best when

  - there are a large number of arguments (difficult to remember their order)

  - some arguments have default values (want to specify only a subset)

- Keyword arguments can be passed in any order

```scss
$color: #F0F; // purple                          SCSS


#d1 {
  font: 24pt bold monospace;
  color: change-color($color, $blue: 0, $green: 256);
}
```

> $color is a positional argument;
> $blue and $green are a keyword arguments;
> results in yellow

Imports

- Extends CSS **@import** so Sass files can import others

  - regardless of which syntax it uses (SCSS or Sass)

- "Partials"

  - Sass files only meant to be imported into others

  - starting filename with an underscore tells Sass
    not to generate a **.css** files from it

- **@import "*file-name*";**

  - don't need to specify file extension or leading underscore for partials

  - looks in current directory by default

  - can specify other locations with **--load-path** command-line option

  - can import more than one file
    with multiple **@import** statements or
    listing multiple filenames separated by commas

Selector Inheritance

- Adds all properties in one rule to another

- Inside destination rule, add one or more
  @extend {*source-rule-selector*};

> must be a "single element" selector, for example, "**td**", but not "**#foo td**"

- Can extend a rule that extends another rule

- Properties defined later take precedence

```
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="extend.css"/>
  </head>
  <body>
    <div class="warning">The sky is falling!</div>
    <div class="error">Goodbye cruel world!</div>
  </body>
</html>
```

```scss
.warning {                          SCSS
  border: solid black 3px;
  background-color: orange;
  color: white;
  font: 18pt sans-serif;
  padding: 10px;
  width: 250px;
}
.error {
  @extend .warning;
  background-color: red;
}
```

The sky is falling!

Goodbye cruel world!

> gets complicated when an extended selector appears in other multi-element selectors (combinatorial explosion) (see reference section on "Merging Selector Sequences")

Mixins

- Reuse lines of Sass

- Optionally parameterized

  - parameters are variable names with optional default values

- **@mixin**

  - defines content

  - can **@include** other mixins

- **@include**

  - includes content

  - can use positional and/or keyword arguments

```scss
@mixin transition($property, $duration, $function: ease-in-out) {
  -moz-transition: $property, $duration, $function;
  -ms-transition: $property, $duration, $function;
  -o-transition: $property, $duration, $function;
  -webkit-transition: $property, $duration, $function;
  transition: $property, $duration, $function;
}


div {
  border: solid red 2px;
  width: 600px;
  @include transition(all, 3s);
}
```

default value

This example demonstrates making it easier to use CSS3 experimental properties that differ across current browsers.

Control Directives

- Mainly used inside mixins and custom functions

- Conditional logic

  - `@if`, `@else if` and `@else`

- Iteration

  - `@for ${var} from {start} [to | through] {end} { ... }`

    - `end` is exclusive with `to` and inclusive with `through`

  - `@each ${var} in {list} { ... }`

  - `@while {condition} { ... }`

    - can reassign variables inside body to affect condition

Custom Functions

- Typically used to compute property values

- Defined in two ways

  - **@function**

  - adding Ruby methods to the module **Sass::Script::Functions**

```
@function season-color($season) {
  @if $season == spring {
    @return green;
  } @else if $season == summer {
    @return red
  } @else if $season == fall {
    @return brown
  } @else if $season == winter {
    @return white
  } @else {
    @return black
  }
}


$season: summer;


#d1 {
  background-color: season-color($season);
  height: 100px;
  width: 100px;
}
```

Recommendations

- Change file extension of existing `.css` files to `.scss`

- Replace repeated property values with variables

- Replace repeated property lines in rules with mixins or selector inheritance

- Use Compass mixins for setting common sets of properties

  - Compass is a framework built on top of Sass

    - adds predefined mixins, CSS property reset and
      support for Blueprint framework that provides grid layout and more

  - ex. CSS3 experimental properties that support border radii, transitions and transforms