IS 2560: Web Tech & Standards Course Introduction

Graduate Program Information Science and Technology

School of Information Sciences

University of Pittsburgh

Alawya Alawami Ph.D.

HTML5 (Part 2)

Why HTML5?

- *Separate presentation from content*
- *Support XHTML and HTML4*
- *Rules*
  - New features should be based on HTML, CSS, DOM, and JavaScript
  - Reduce the need for external plugins
  - Better error handling
  - More mark up to replace scripting
  - HTML5 should be device independent
  - Dev process should be visible to the public

Remember

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

HTML5 definition: `<!DOCTYPE html>`

Minimal Document

XHTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
 <html xmlns="http://www.w3.org/1999/xhtml">
<head>
                <title></title>
</head>
<body>

</body>
</html>
```

HTML5

```
<!DOCTYPE html >
 <html >
<head>
                <title></title>
</head>
<body>

</body>
</html>
```

HTML5: Detection

When your browser renders a web page, it constructs a Document Object Model, a collection of objects that represent the HTML elements on the page. Every element is represented in the DOM by a different object.

In browsers that support HTML5 features, certain objects will have unique properties. A quick peek at the DOM will tell you which features are supported.

HTML5: Detection

[Modernizr] is an open source, MIT-licensed JavaScript library that detects support for many HTML5 & CSS3 features.

Npm install

npm **install -g** modernizr

Download custom javascript

https://modernizr.com/download?setclasses

HTML5: Detection

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>HTML5 sure is fun</title>
  <script src="modernizr.min.js"></script>
</head>
<body>
  ...
</body>
</html>
```

HTML5: Detection

Modernizr runs automatically. When it runs, it creates a global object called Modernizr, that contains a set of Boolean properties for each feature it can detect. For example, if your browser supports the canvas API, the Modernizr.canvas property will be true – otherwise the property will be false.

HTML5: Detection

```
if (Modernizr.canvas) {
  // let's draw some shapes!
} else {
  // no native canvas support available :(
}
```

More on HTML5 detection (and Modernizr) here: http://diveintohtml5.org/detect.html

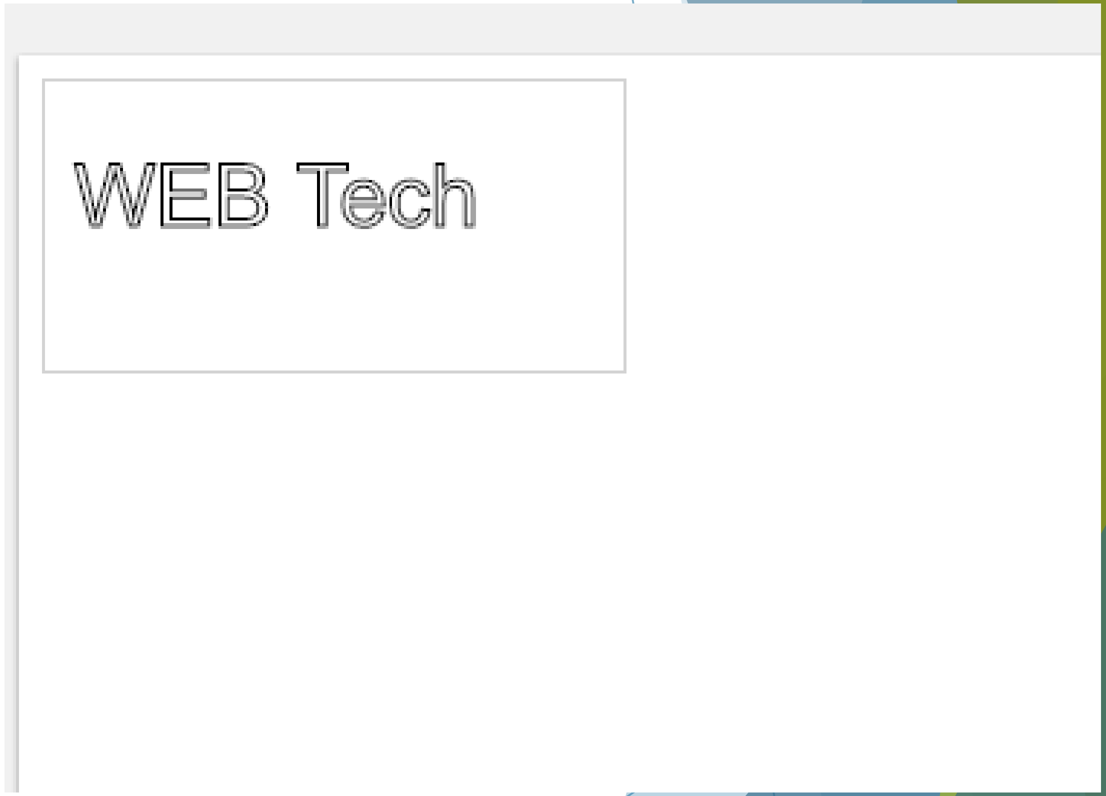
HTML5 Canvas

- It is your free form drawing area.
- Define area

<canvas id="myCanvas" width="200" height="100“ style="border:1px solid #d3d3d3;">

- The canvas element has no drawing abilities of its own. All drawing must be done inside a JavaScript:
- Properties
  - Look them up here
- Can You draw INFSCI2560 inside your Canvas?

```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="200" height="100"
style="border:1px solid #d3d3d3;">
Your browser does not support the canvas element.
</canvas>

<script>
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.font = "30px Arial";
ctx.strokeText("WEB Tech",10,50);
</script>

</body>
</html>
```



Local Storage

- HTML local storage provides two objects for storing data on the client:
  - *window.localStorage* - stores data with no expiration date
  - *window.sessionStorage* - stores data for one session (data is lost when the tab is closed)

The localStorage Object

- The **localStorage** object stores the data with no expiration date.
- The data will not be deleted when the browser is closed.

```
// Store
localStorage.setItem("lastname", "Smith");
// Retrieve
document.getElementById("result").innerHTML = localStorage.getItem("lastname");
```

The localStorage Object

- Previous example could also be written like this:

```
// Store
localStorage.lastname = "Smith";
// Retrieve
document.getElementById("result").innerHTML = localStorage.lastname;
```

- Removing item from localStorage:

```
localStorage.removeItem("lastname");
```

The sessionStorage Object

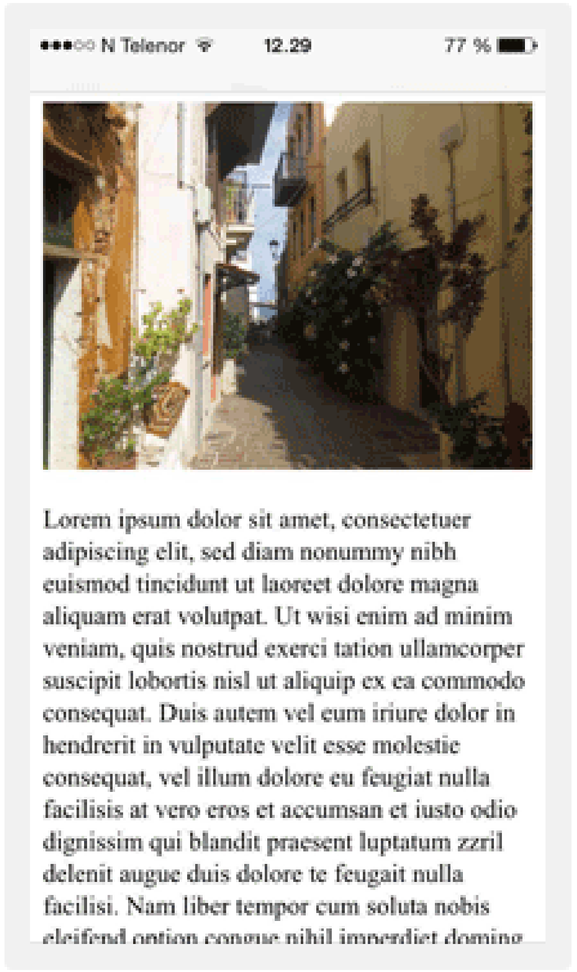
- The sessionStorage object stores the data for only one session. The data is deleted when the user closes the browser window.

```
if (sessionStorage.clickcount) {
    sessionStorage.clickcount = Number(sessionStorage.clickcount) + 1;
} else {
    sessionStorage.clickcount = 1;
}
document.getElementById("result").innerHTML = "You have clicked the button " +
sessionStorage.clickcount + " time(s) in this session.";
```

HTML Viewport

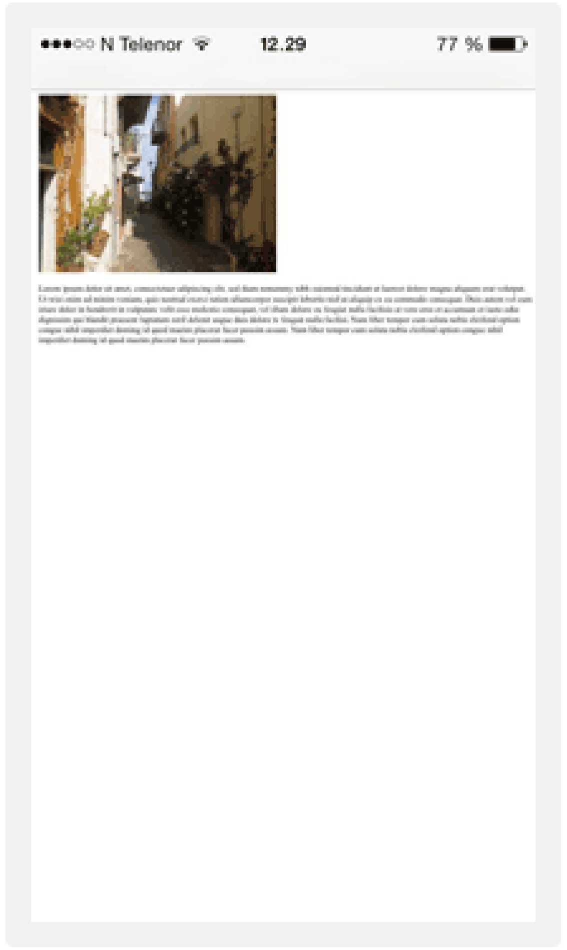
- <meta name="viewport" content="width=device-width, initial-scale=1.0">

- The viewport is the user's visible area of a web page.
- It varies with the device, and will be smaller on a mobile phone than on a computer screen.



Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming

With the viewport meta tag



Without the viewport meta tag

More HTML Syntax

- https://www.w3schools.com/html/html5_syntax.asp
- Validator

  https://validator.w3.org

SVG

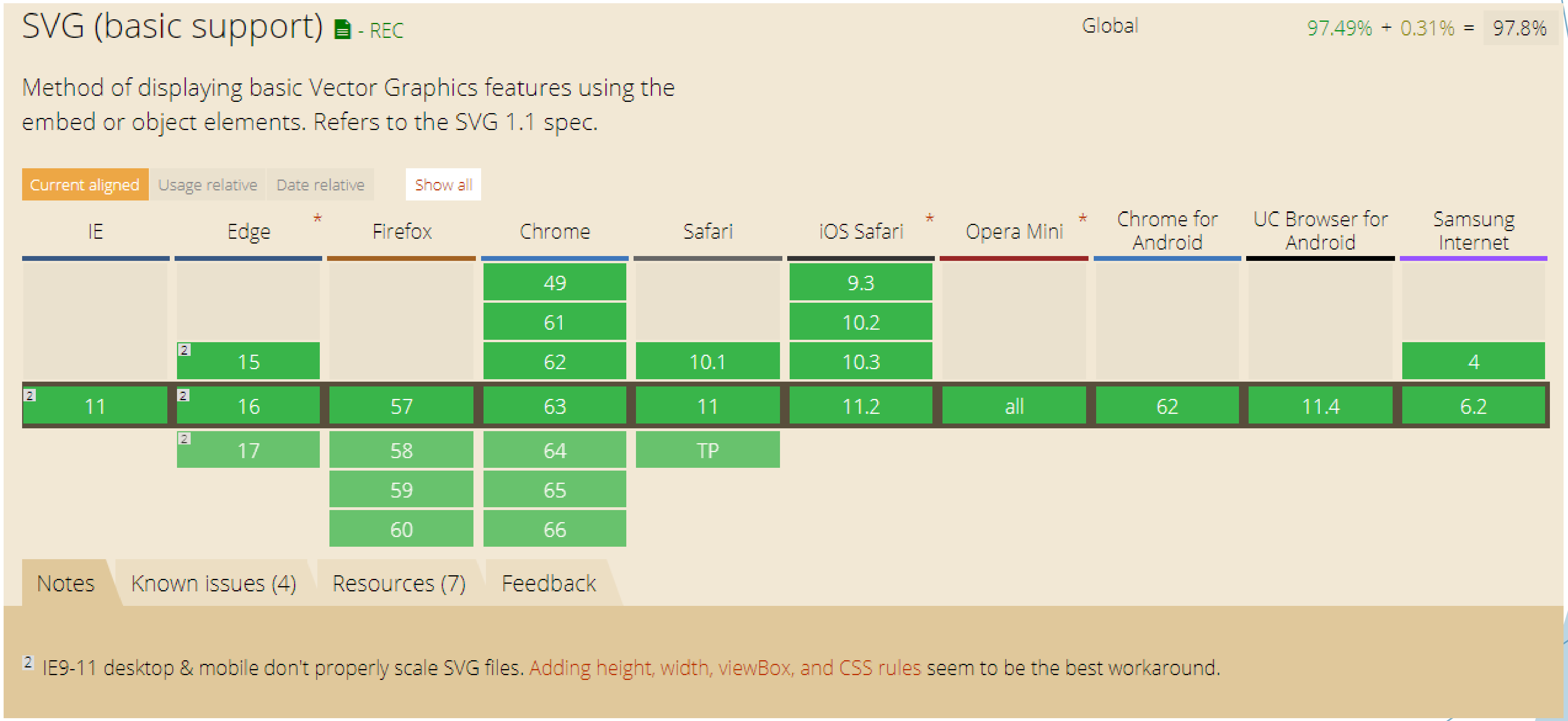
Overview

- SVG
- Simple Primitives
  - Attributes
- Text and Path
- ViewBox
- Filters and Gradients

What is SVG

- SVG or Scalable Vector Graphics is a way to define scalable graphical objects using XML
- SVG competes with Flash (or the other way around) and becomes a piece of HTML 5
- SVG makes it possible to display resizable high quality graphical images in a browser
- One problem with SVG is that IE does not currently support it – but will eventually.

Web Tech Support



SVG (basic support) - REC

Global 97.49% + 0.31% = 97.8%

Method of displaying basic Vector Graphics features using the embed or object elements. Refers to the SVG 1.1 spec.

Current aligned | Usage relative | Date relative | Show all

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49		9.3				
			61		10.2				
	[2] 15		62	10.1	10.3				4
[2] 11	[2] 16	57	63	11	11.2	all	62	11.4	6.2
	[2] 17	58	64	TP					
		59	65						
		60	66						

Notes | Known issues (4) | Resources (7) | Feedback

[2] IE9-11 desktop & mobile don't properly scale SVG files. Adding height, width, viewBox, and CSS rules seem to be the best workaround.

https://caniuse.com/#feat=svg

A Simple Circle

```
<?xml version="1.0" standalone="no"?>

<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
   "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg width="100%" height="100%" version="1.1"
   xmlns="http://www.w3.org/2000/svg">

<circle cx="100" cy="50" r="45" stroke=“yellow"
   stroke-width="2" fill=“blue"/>

</svg>
```

Some SVG Primitives

- The basic primitives in SVG include:
  - Rectangle <rect>
  - Circle <circle>
  - Ellipse <ellipse>
  - Line <line>
  - Polyline <polyline>
  - Polygon <polygon>
  - Path <path>

Attributes Common to Most Objects

This is only a subset of the attributes available

- CSS and XML
  - Attributes used for styling and identifications of element
    - id, style,class
- FillStroke
  - Attributes for the filling (placing color inside the object) and stroking (drawing the lines around the object) of vector objects.
    - fill, fill-opacity, fill-rule
    - stroke, stroke-width, stroke-opacity
    - stroke-dasharray, stroke-dashoffset
    - stroke-linecap, stroke-linejoin, stroke-miterlimit

Attributes for Path

- Just about any kind of path can be created using commands which include:
  - M = moveto
  - L = lineto,   H = horizontal lineto,   V = vertical lineto
  - C = curveto,   S = smooth curveto
  - Q = quadratic Bézier curve
  - T = smooth quadratic Bézier curveto
  - A = elliptical Arc
  - Z = closepath
- When caps are used positions are absolute, lower case implies relative.
- Two simple examples are a horizontal line and a quadratic curve:
  - <path hline = “M100 100 l100 0”/>
  - <path qline = “M100 100 q150 300 300 0” />

Sample Attributes for Circle

cx	The x-axis center of the circle
cy	The y-axis center of the circle
r	The circle's radius

Example

```
<circle cx="100" cy="100" r="90" fill="blue" />
```

Sample Attributes for Text

x	A list of x-axis positions. The nth x-axis position is given to the nth character in the text. If there are additional characters after the positions run out they are placed after the last character. (0 default)
y	A list of y-axis positions. See 'x' description. (0 default)
dx	A list of lengths which moves the characters relative to the absolute position of the last glyph drawn. (see x)
dy	A list of lengths which moves the characters relative to the absolute position of the last glyph drawn. (see x)
rotate	A list of rotations. The nth rotation is performed on the nth character. Additional characters are NOT given the last rotation value.
textLength	A target length for the text that the SVG viewer will attempt to display the text between by adjusting the spacing and/or the glyphs. (default: The text's normal length)
lengthAdjust	Tells the viewer what to adjust to try to accomplish rendering the text if the length is specified. The two values are 'spacing' and 'spacingAndGlyphs'.

Path and Text

- Text is normally drawn as one would expect, but it can be made to follow any path.
- For example:

```
<defs>
  <path id="path1" d="m10,10 q65,200 130,0 " />
 </defs>
 <text x="10" y="100" >
     <textPath xlink:href="#path1">
     I love the School of Information Science
     </textPath>
 </text>
```

SVG and HTML

- An SVG document can stand alone or be placed into another document.
- A variety of techniques, use of embed, object, or iframes can be used and each has advantages and disadvantages
- The current simplest method is simply to use and SVG element in the HTML document with a default local namespace

```
<html>
<body>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <circle cx="100" cy="50" r="40" stroke="black"
  stroke-width="2" fill="red"/>
</svg>
</body>
</html>
```

SVG: Viewports and ViewBox

- Basically, when an SVG element is specified, its width and height are specified and define the default coordinate system with 0,0 being the upper left corner.
  - <svg width="200" height="200">
- Graphics drawn would use coordinates between 0,0 and 200,200 or be clipped.
- To change the coordinate units, specify a viewBox attribute. Things can get complicated, but to use decimals from 0 to 1, you could specify:
  - <svg width="200" height="200” viewBox = “0 0 1 1”>

SVG Filters and Gradients

- Filters are effects that may be applied to a given element.
  - There are more than a dozen filters ranging from blurs, convolutions and compositions, tiling, lighting of various forms
- Gradients are of two forms linear and radial
  - Linear gradients basically allow a one dimensional gradient
  - Radial gradients operate in two dimensions
- Both filters and gradients are associated with an element via an id

Simple Gradient

```
<svg width="100%" height="100%"
version="1.1“ xmlns="http://www.w3.org/2000/svg">

<defs>
<radialGradient id="gb" cx="50%" cy="50%" r="50%“ fx="50%" fy="50%">
<stop offset="0%" style="stop-color:rgb(200,200,200); stop-opacity:0"/>
<stop offset="100%" style="stop-color:rgb(0,0,255); stop-opacity:1"/>
</radialGradient>
</defs>

<ellipse cx="230" cy="200" rx="110" ry="100“ style="fill:url(#gb)"/>

</svg>
```

SVG Demo

- http://www.pitt.edu/~aaa221/Demo/SVG/demo.html

IS 2560: CSS

Graduate Program Information Science and Technology

School of Information Sciences

University of Pittsburgh

Overview

- Context
- Overview of Rendering
- Basic format of a CSS style sheets
- Examples of xhtml and xml uses of CSS
- Cascading Style Sheets Features
  - CSS Version 1
  - CSS Version 2
  - CSS Version 3
- Additional considerations in the use of style sheets

Context

- While html document elements have default appearance characteristics set in browsers, XML documents have no predefined presentation and require that these be defined.
- Some elements in html – div and span – have no predefined appearance
- Additionally, many designers want better control over document appearance and presentation.
- The XML Style Language (XSL) provides a capability to style XML documents, but many users prefer the Cascading Style Sheets standards (1,2, and 3) which work for html, xhtml, and xml documents.
- This slide set addresses the very basic capabilities of CSS.

The Evolution of Rendering in Browsers

- Initially, the rendering of HTML elements was built into the browsers
- With the development of XML, the focus returned to the goal of device independent formatting
- Cascading Style Sheets were the first step in this direction. They work for both HTML and XML
- Cascading Style Sheets worked differently in Internet Explorer and Netscape Navigator
- Cascading Style Sheets have evolved and are now specified at three levels.
- The Formatting Object specification (XSLT/FO) initially took the rendition of XML documents a step beyond CSS, but has generally been ignored while CSS has grown in capability.

Rendering Generally

- Historically, rendering to different devices required radical adjustments – i.e., rendering to a teletype was different than rendering to a phototypesetter (n.b. Scribe)
- With the development of “all-points addressable devices” – laser printers and bit-mapped screens – the need to have multiple rendering engines has decreased.
- Today, with the need to audio-presentation, graphical presentation, and other semantically differentiated output, the need for device independent rendering has returned to some extent and is met by media queries in CSS2 and 3.

Graphical Rendering

- To understand graphical rendering, one needs to go back to Scribe and Interpress technologies.
- Consider the following page design goals:
  - The design of the first page of a chapter or letter is different from the succeeding pages
  - The design of left and right pages should be different
  - Material in the header or footer of a page should contain both static and dynamic information
  - Some material should be “flowed” differently – text fills from the top while footnotes fill from the bottom
  - Material should inherit some relative values from parents.
- CSS and XSL/FO provide these and additional capabilities.

Cascading Style Sheets(CSS)

- The style sheet associates attributes with elements:
  - The nature of the element -- block, inline, list item, none
  - The nature of the text --font characteristics, text alignment, background, etc.
- Styles can be specified multiple ways:
  - With HTML, it can be placed in the header section of the document:

```
<style type = “text/css”>
…
</style>
```

  - Or associated with an html document using a link:

```
<LINK href="special.css" rel="stylesheet" type="text/css">
```

  - Or it can be embedded using a fragment reference to the id of an internal element

Basic Syntax

- The basic syntax for a style sheet id as follows:
  - Element {property: value; property: value;}
  - H1 {color:green; font-size: 20pt;}
- There are complex rules that can further the sophistication of the style. For example if elements have class attributes, element styles can be differentiated:
  - Element.CAttrib {p:v; p:v;}
- This style will only be applied when the element is of the class CAttrib
  - <h1 class = “xyz”>Will be impacted</h1>
- Further, a class style can be associated with deifferent elements that have the same class attribute
  - .CAttrib {p:v; p:v;}
- Etc. ad nauseaum – see next slide
- NB Firefox is good for examining style inheritance

Further Differentiating Styles

- CLASS and ID attributes may be used to differentiate elements
  - ELEMENT.CLASS {style info}
  - ELEMENT#CLASS {style info}
- Multiple Elements may use the same style
  - E1, E2, E3 {style info}
- The first letter and first line may be handled separately
  - ELEMENT:first-letter {style info}
  - ELEMENT:first-line {style info}
- Elements may inherit from multiple overlapping styles
  - E1, E2, E3, E4 {some style info}
  - E3 {some style info}
  - E1, E2 {some style info}
- Elements may be differentiated by context(parent)
  - ELEMENT1 {style info}
  - PARENTB ELEMENT1 {style info}
- Elements may be differentiated individually
  - <ELEMENT1 STYLE="style info">

CSS reference

- https://www.w3schools.com/css/default.asp

Why Cascading?

Why Cascading

- Cascading Style Sheets “cascade” in several ways
- Origin of a style
  - Styles imported from files are the base
  - Styles in the head “cascade” over them
  - Inline styles are most definitive
- Styles also “cascade” from general to specific
  - Parent elements styles cascade to children
  - Element styles can be specialized
    - Class styles override element styles
    - ID styles override element and class styles
  - When a style specifies parent and child it dominates simple element style

An xhtml Example

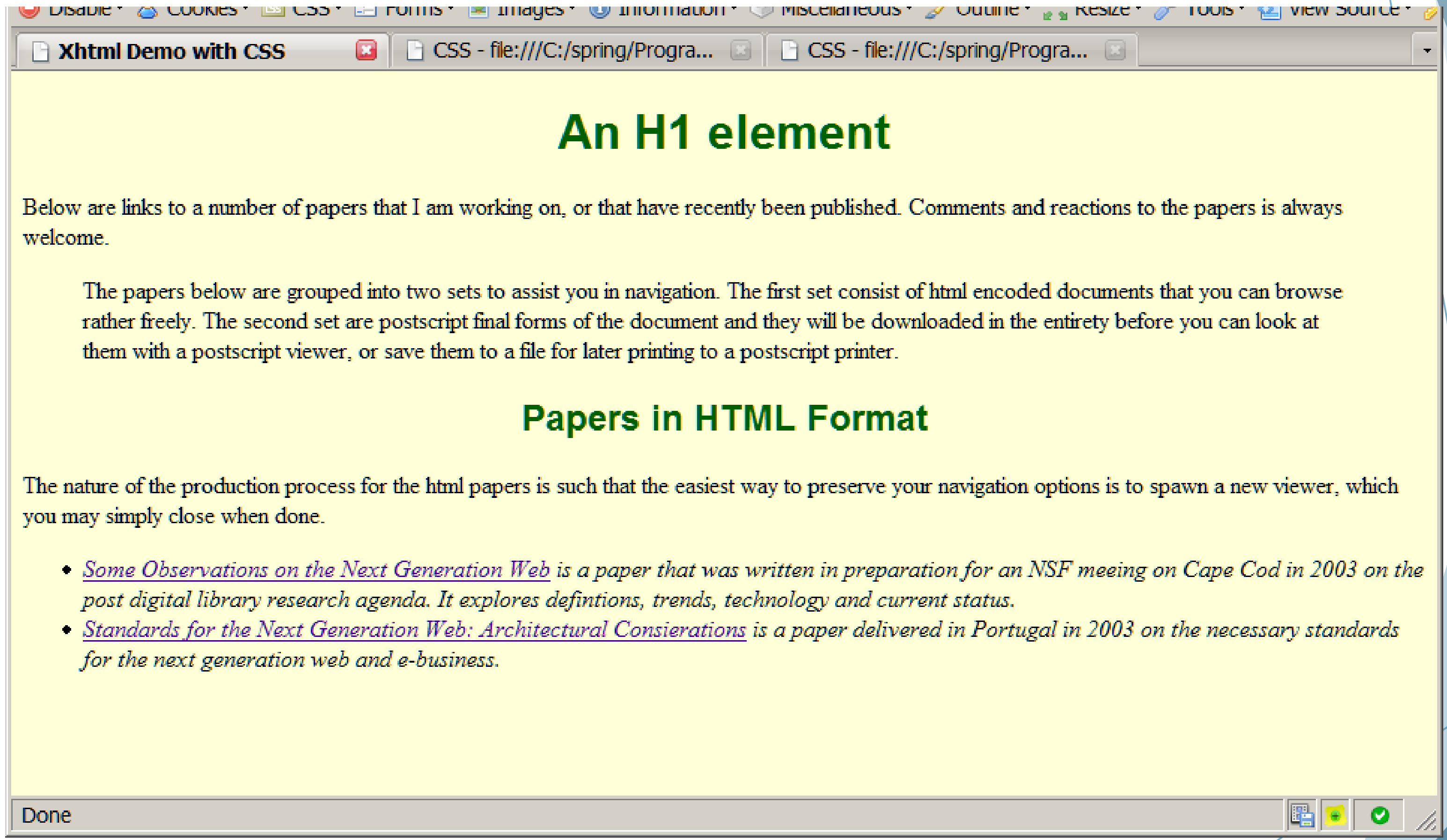
- Given the following Document

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html    PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
         "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
         <title>Xhtml Demo with CSS</title>
         <link rel="stylesheet" type="text/CSS" href="a.css" />
  </head>
  <body>
<h1>An H1 element</h1>
<p>Below are welcome. </p>
<h2>Papers in HTML Format</h2>
<p>The nature of the production process for the html papers is such when done. </p>
<ul>
<li><a HREF="papers/PrepObserv_Final.htm" target="right">Some Observations on
the Next Generation Web</a> is a defintions, trends, technology and current status. </li>
<li><a HREF="papers/SEB.htm" target="right">Standards for the
2003 on the necessary standards for the next </li>
</ul>
<body>
</html>
```

- And the following stylesheet

```
body{background: #FFFFD8;
        margin-top: 20;}

a:link     {color: #400080;
        background: #FFFFD8
       font-style: bold;}

h1, h2    {font-weight: bold;
        text-align: center;
        color: #006000;
        background: #FFFFD8;
        font-family: Arial, 'Gill Sans',  sans-serif;}

ul, ol{font-style: italic;}
```

Firefox Output



An H1 element

Below are links to a number of papers that I am working on, or that have recently been published. Comments and reactions to the papers is always welcome.

The papers below are grouped into two sets to assist you in navigation. The first set consist of html encoded documents that you can browse rather freely. The second set are postscript final forms of the document and they will be downloaded in the entirety before you can look at them with a postscript viewer, or save them to a file for later printing to a postscript printer.

Papers in HTML Format

The nature of the production process for the html papers is such that the easiest way to preserve your navigation options is to spawn a new viewer, which you may simply close when done.

- *Some Observations on the Next Generation Web is a paper that was written in preparation for an NSF meeing on Cape Cod in 2003 on the post digital library research agenda. It explores defintions, trends, technology and current status.*
- *Standards for the Next Generation Web: Architectural Consierations is a paper delivered in Portugal in 2003 on the necessary standards for the next generation web and e-business.*

An xml Example

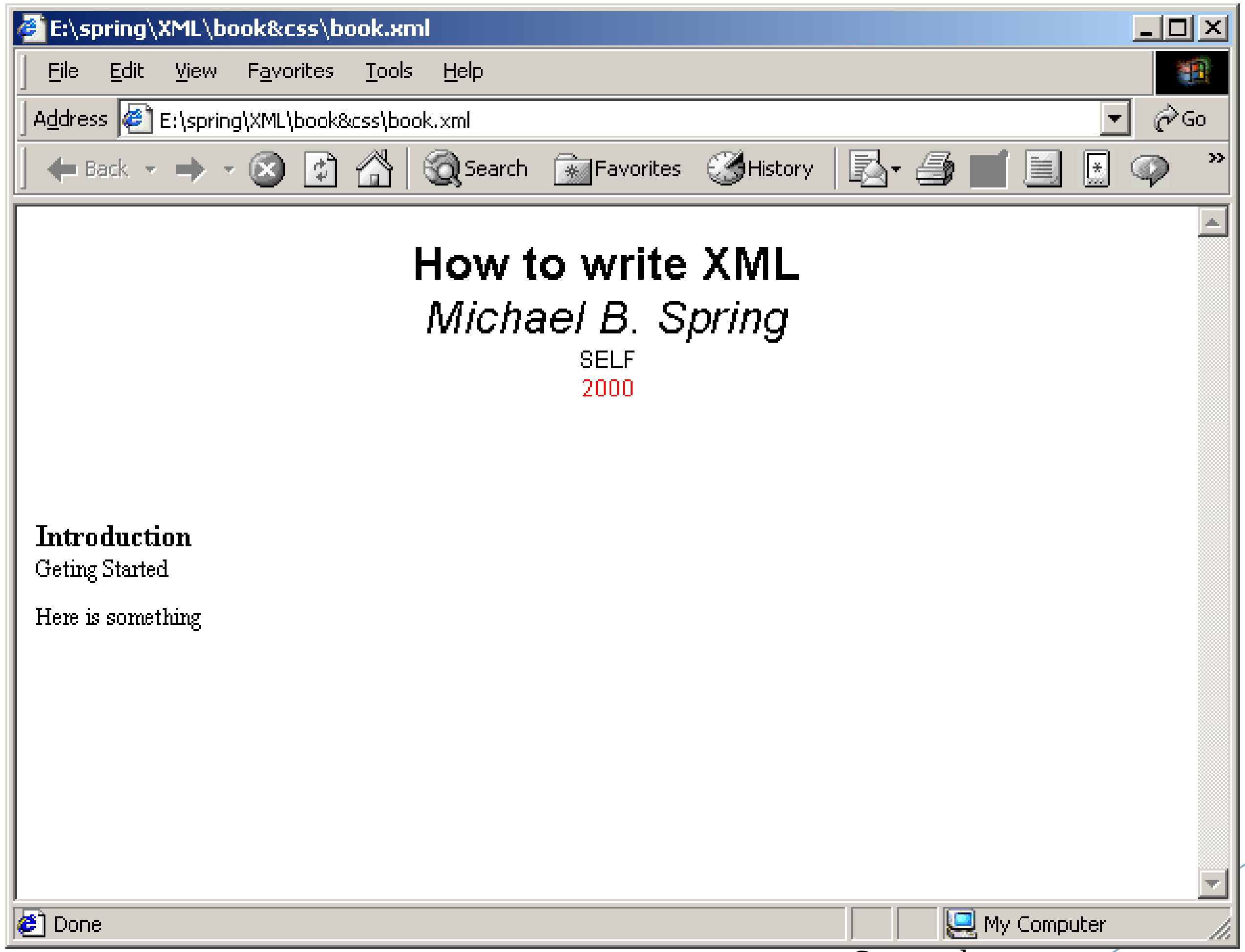
- Given the following Document

```
<?xml version='1.0' standalone='no'?>
<?xml-stylesheet type="text/css" href="book.css"?>
<book>
<titlepage>
<btitle>How to write XML</btitle>
<author>Michael B. Spring</author>
<publisher>SELF</publisher>
<date>2000</date>
</titlepage>
<chapter>
<ctitle>Introduction</ctitle>
<para>Geting Started</para>
<para>Here is something</para>
</chapter>
</book>
```

- And the following stylesheet

```
book { display: block }
titlepage { display: block; font-family: Arial;
      text-align: center; font-size: 24}
btitle { display: block; font-weight: bold }
author { display: block; font-style: italic }
publisher { display: block; font-size: 12 }
date { display: block; font-size: 12;
      color: red }
chapter { display: block; font-family: Times;
      margin-top: 60px; font-size: 12}
ctitle { display: block; font-size: 16;
      font-weight: bold }
para { display: block;
      margin-bottom: 10px }
```

Internet Explorer Output


E:\spring\XML\book&css\book.xml
File Edit View Favorites Tools Help
Address E:\spring\XML\book&css\book.xml
Go
Back
Search
Favorites
History
Done
My Computer

How to write XML

Michael B. Spring

SELF

2000

Introduction

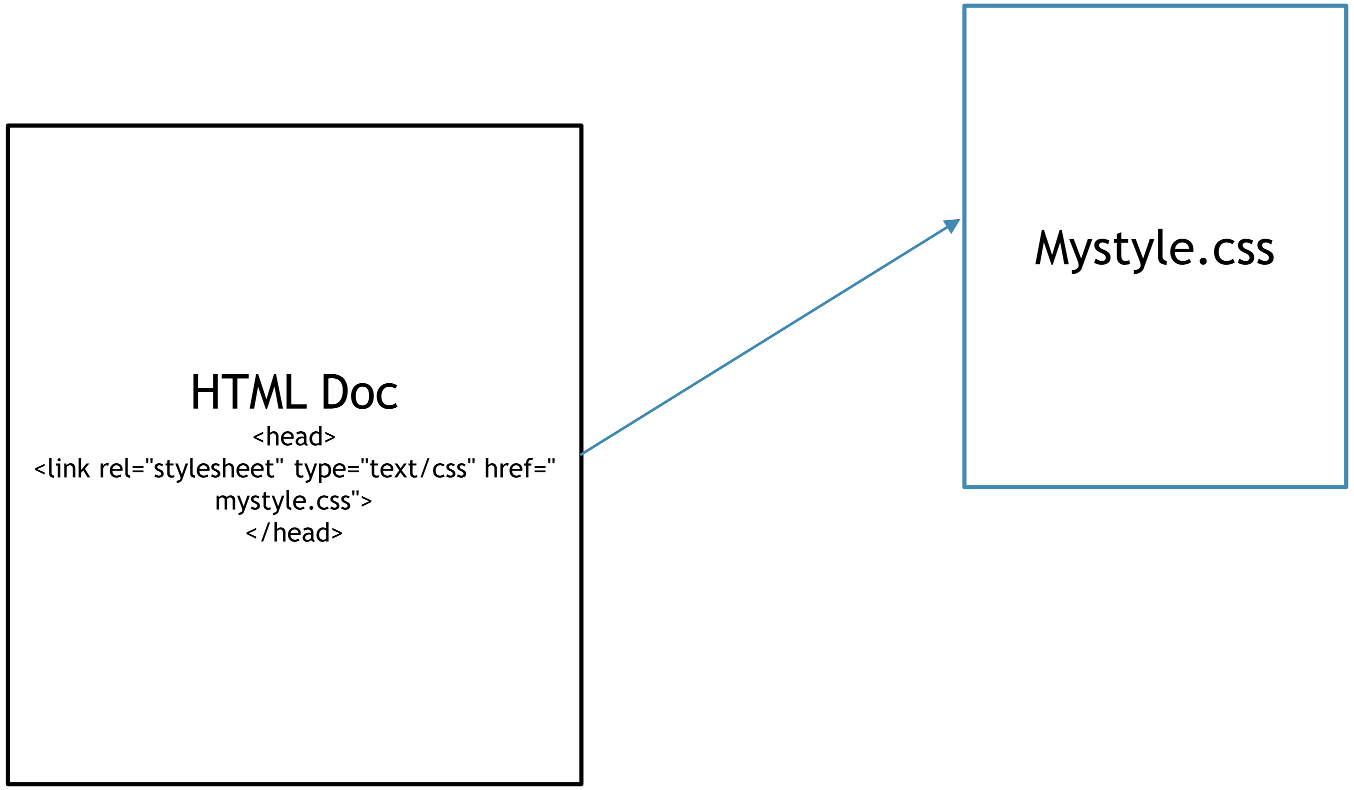
Geting Started

Here is something

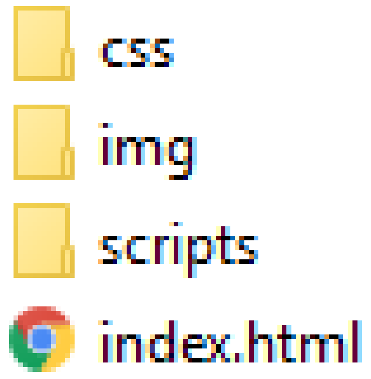
Strategies for Styles

- The simplest style strategy is to associate styles with existing xml or xhtml elements. This is a good strategy when using xml.
    - h1 {style info}
- Using class attributes for elements allows one to define styles for the classes. This allows one to distinguish between an h1.chapter-title and an h1.section-title allowing for more semantic information:
    - .chapter-title {style info}
- ID's are frequently used to specify style information for special page elements – e.g. elements that will be filled with dynamic content
- Finally, increasingly styling information is specified in relative terms – percentages rather than pixels and ems rather than points.

Structure so far



```
<head>
<link rel="stylesheet" type="text/css" href="
mystyle.css">
</head>
```

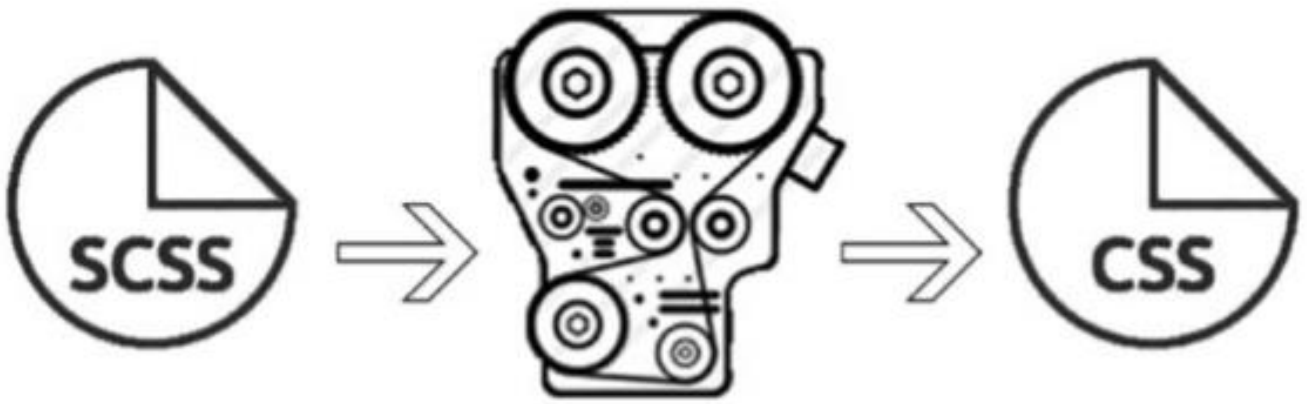


CSS Preprocessors

Brief intro

CSS Preprocessor

- A preprocessor takes one form of data and converts it to another
- Some of the popular
  - Sass (Syntactically Awesome Stylesheets)
  - Compass (COMPrehensive ASSembler)
  - Less (Leaner SS)



Why CSS Preprocessor?

- CSS can get too complicated, too long, hard to maintain
- Preprocessors
  - Tools for development
  - Language → compilation → CSS
  - Gives more functionality to CSS
  - Dynamic style sheets
  - CSS properties as variables
  - Fewer lines of code

Less

- Less is a tool that extends CSS functionality
- Originally build with Ruby, now it is JavaScript

=> Not written in any other language(*Not PHP*)

Usage

- Use with Node.js:

```
>npm install -g less
> lessc styles.less styles.css
```

- Or the browser:

```
<link rel="stylesheet/less" type="text/css" href="styles.less" /
<script src="//cdnjs.cloudflare.com/ajax/libs/less.js/3.7.1/less.min.js"
></script>
```

- More info

Features

- Variables allow you to specify widely used values in a single place, and then re-use them throughout the style sheet

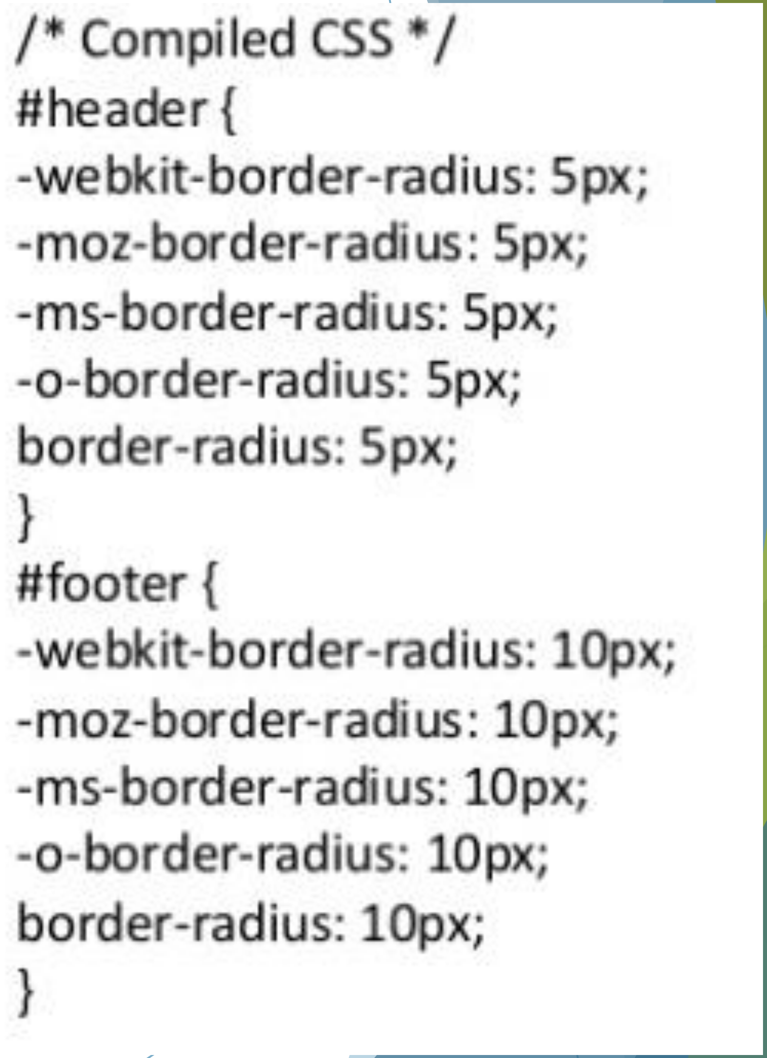
Make changing the value easier

- Variables in LESS, use the @ symbol
  - @maincolor: #232323
  - Global styles
  - Define once, use anywhere
  - Change once, update everywhere

Mix-ins

- Classes injected into other classes
- Reusable snippets of code
- Optional parameters (act likes a function)

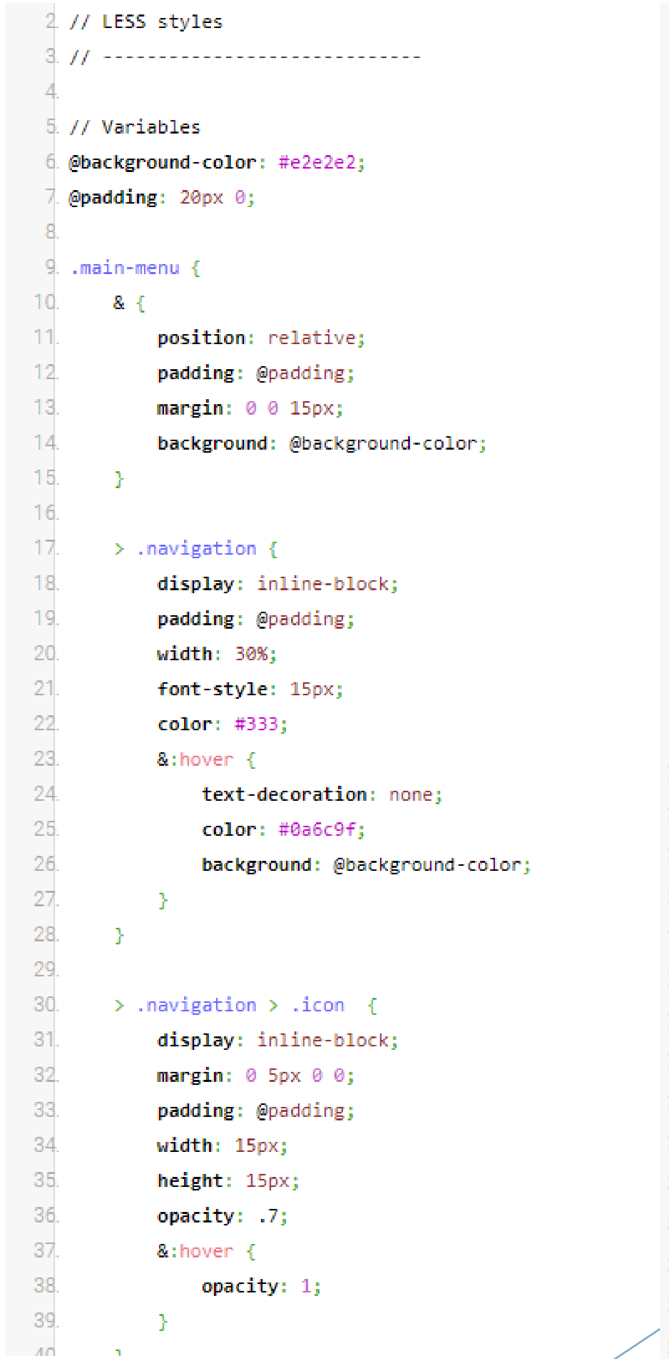
```
// LESS
.rounded-corners (@radius: 5px) {
        -webkit-border-radius: @radius;
        -moz-border-radius: @radius;
        -ms-border-radius: @radius;
        -o-border-radius: @radius;
        border-radius: @radius; }

#header { .rounded-corners; }
#footer { .rounded-corners(10px); }
```



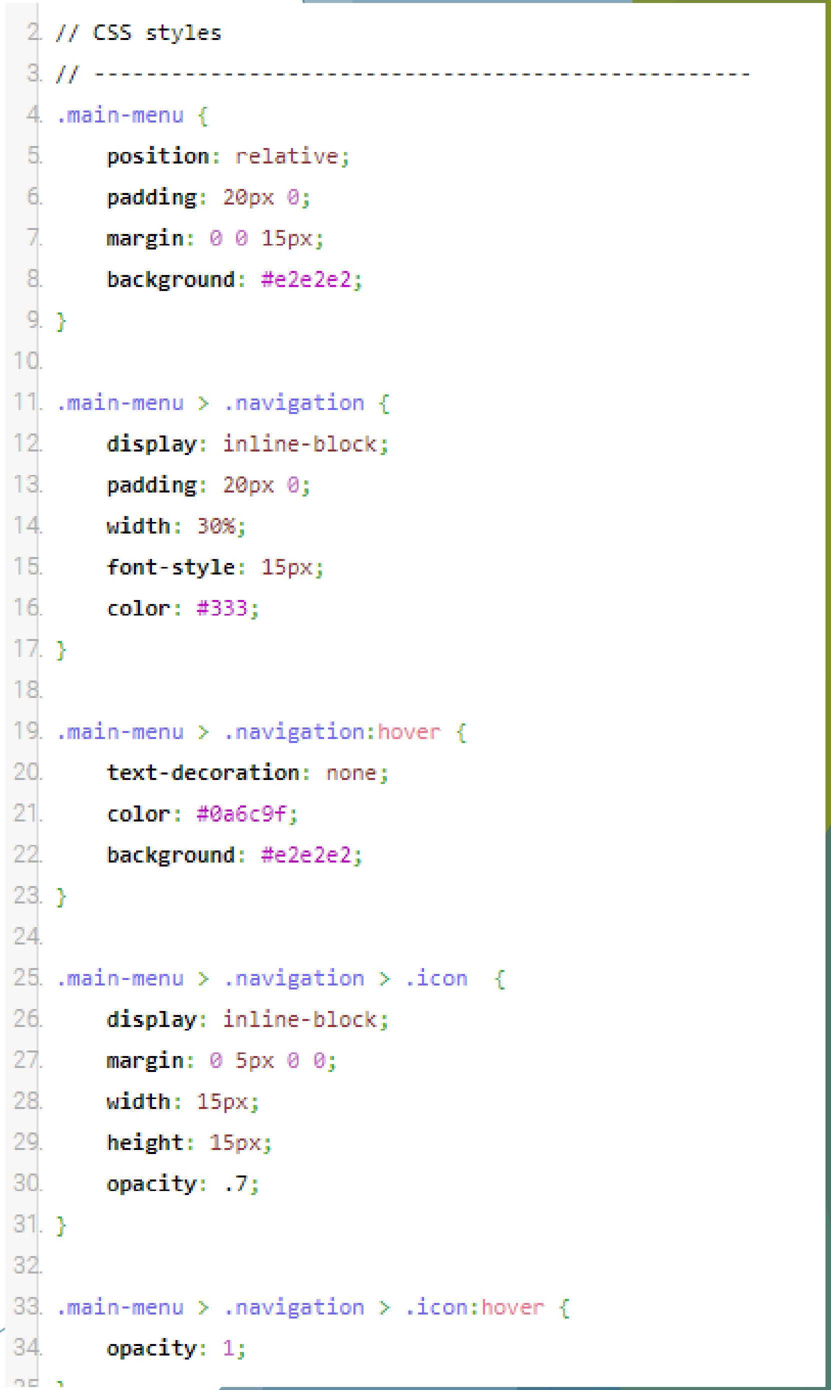
```
/* Compiled CSS */
#header {
-webkit-border-radius: 5px;
-moz-border-radius: 5px;
-ms-border-radius: 5px;
-o-border-radius: 5px;
border-radius: 5px;
}
#footer {
-webkit-border-radius: 10px;
-moz-border-radius: 10px;
-ms-border-radius: 10px;
-o-border-radius: 10px;
border-radius: 10px;
}
```

Nesting

- Gives your CSS hierarchy
- Stack child elements into parent element



```
// LESS styles
// ----------------------------

// Variables
@background-color: #e2e2e2;
@padding: 20px 0;

.main-menu {
    & {
        position: relative;
        padding: @padding;
        margin: 0 0 15px;
        background: @background-color;
    }

    > .navigation {
        display: inline-block;
        padding: @padding;
        width: 30%;
        font-style: 15px;
        color: #333;
        &:hover {
            text-decoration: none;
            color: #0a6c9f;
            background: @background-color;
        }
    }

    > .navigation > .icon  {
        display: inline-block;
        margin: 0 5px 0 0;
        padding: @padding;
        width: 15px;
        height: 15px;
        opacity: .7;
        &:hover {
            opacity: 1;
        }
```



```
// CSS styles
// -----------------------------------------------------
.main-menu {
    position: relative;
    padding: 20px 0;
    margin: 0 0 15px;
    background: #e2e2e2;
}

.main-menu > .navigation {
    display: inline-block;
    padding: 20px 0;
    width: 30%;
    font-style: 15px;
    color: #333;
}

.main-menu > .navigation:hover {
    text-decoration: none;
    color: #0a6c9f;
    background: #e2e2e2;
}

.main-menu > .navigation > .icon  {
    display: inline-block;
    margin: 0 5px 0 0;
    width: 15px;
    height: 15px;
    opacity: .7;
}

.main-menu > .navigation > .icon:hover {
    opacity: 1;
```

Operation

- Operations let you add, subtract, divide and multiply property values and colors
  - giving you the power to create complex relationships between properties.
- Operations should only be performed within parentheses in order to ensure compatibility with CSS.

More resources

- https://brianflove.com/2012/10/03/less-tutorial-and-presentation/
- In-depth guide http://lesscss.org/features/

Intro to Responsive Web Design

Overview

- Definition of the problem
- Nature of the solution
- Examples
- Issues
- Specific techniques

Responsive design

The Problem: Multiple Device Sizes

- A wide array of mobile devices, tablets, and desktops present a challenge for designers.
- Device sizes include:
  - 320 x 480 px: Smartphone
  - 480 x 320 px: Smartphone in landscape orientation
  - 768 x 1024 px: iPad / tablet
  - 1024 x 768 px: iPad in landscape orientation / netbook
  - Anything larger: Desktop / laptop computer

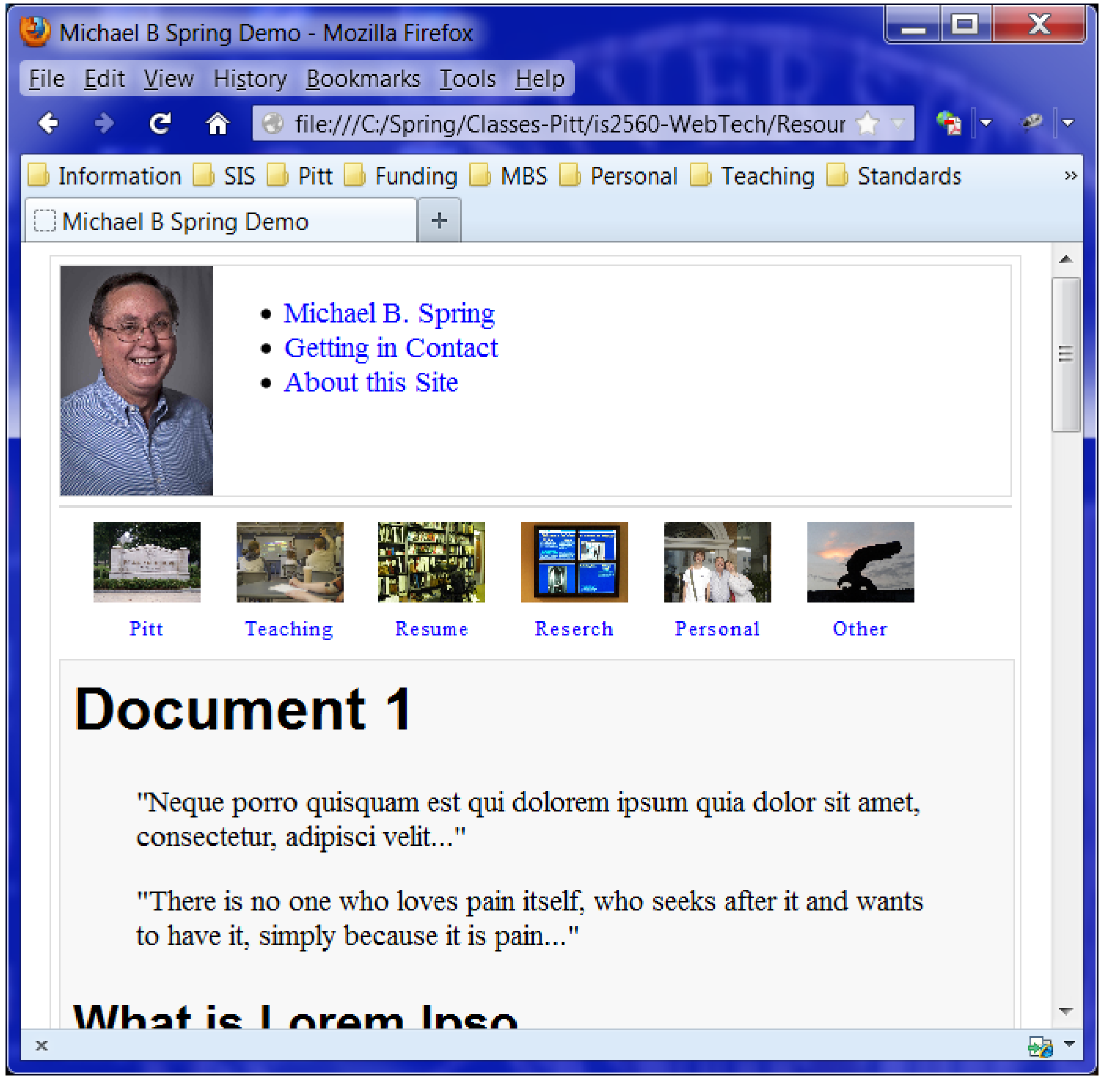
The Solution in a Nutshell

- Responsive Web Design is the name given to a series of techniques and technologies that allow the layout of a web page to adjust to the device screen resolution
- Several techniques are woven together, but the most important relates to CSS media queries that apply styles based on screen size. Very simplistically when the screen size drops to 480px, the font size of H1 elements is set to 24pts and img's of class "icon" are not displayed.

```
@media screen and (max-width: 480px) {
  h1 {font-size: 24px;
  img.iconimage {display: none;}
}
```

- The next page shows the change

Tablet icon menu becomes anchor list



Michael B Spring Demo - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/Spring/Classes-Pitt/is2560-WebTech/Resour
Information SIS Pitt Funding MBS Personal Teaching Standards
Michael B Spring Demo

- Michael B. Spring
- Getting in Contact
- About this Site

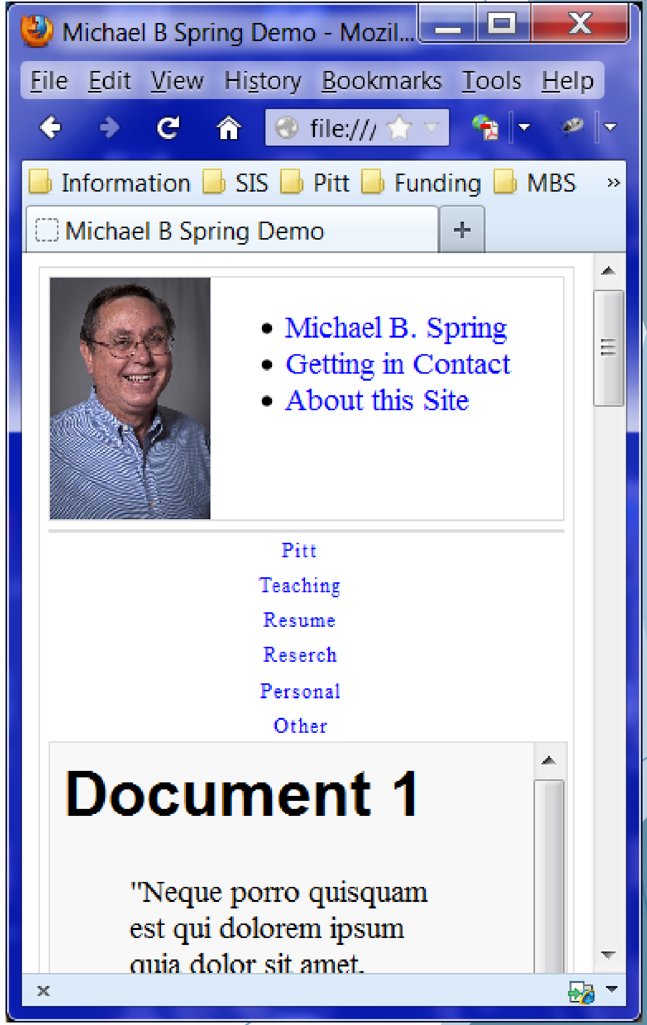
Pitt Teaching Resume Reserch Personal Other

Document 1

"Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit..."

"There is no one who loves pain itself, who seeks after it and wants to have it, simply because it is pain..."

What is Lorem Ipso



Michael B Spring Demo - Mozil...
File Edit View History Bookmarks Tools Help
file:///
Information SIS Pitt Funding MBS
Michael B Spring Demo

- Michael B. Spring
- Getting in Contact
- About this Site

Pitt
Teaching
Resume
Reserch
Personal
Other

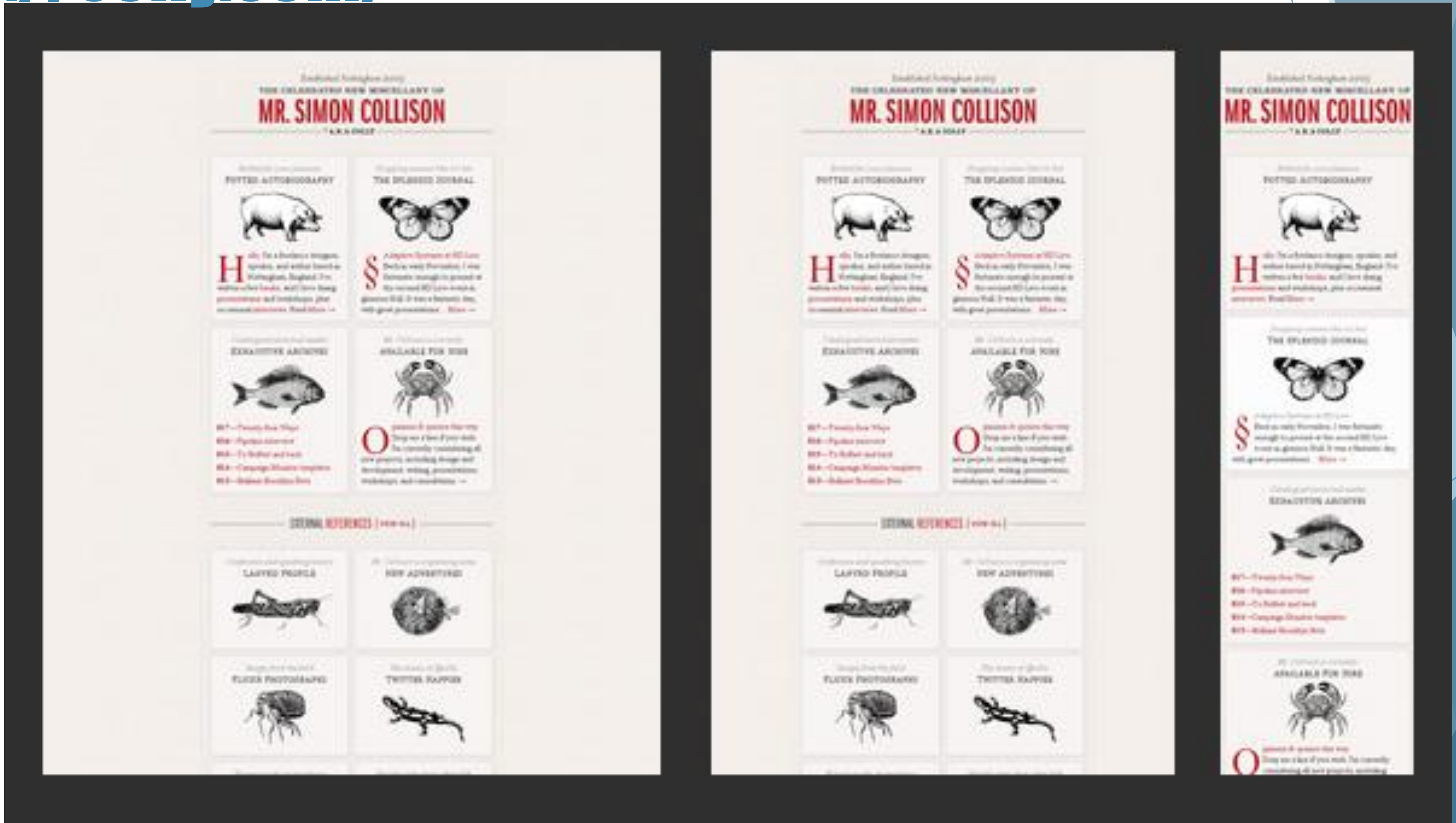
Document 1

"Neque porro quisquam est qui dolorem ipsum quia dolor sit amet.

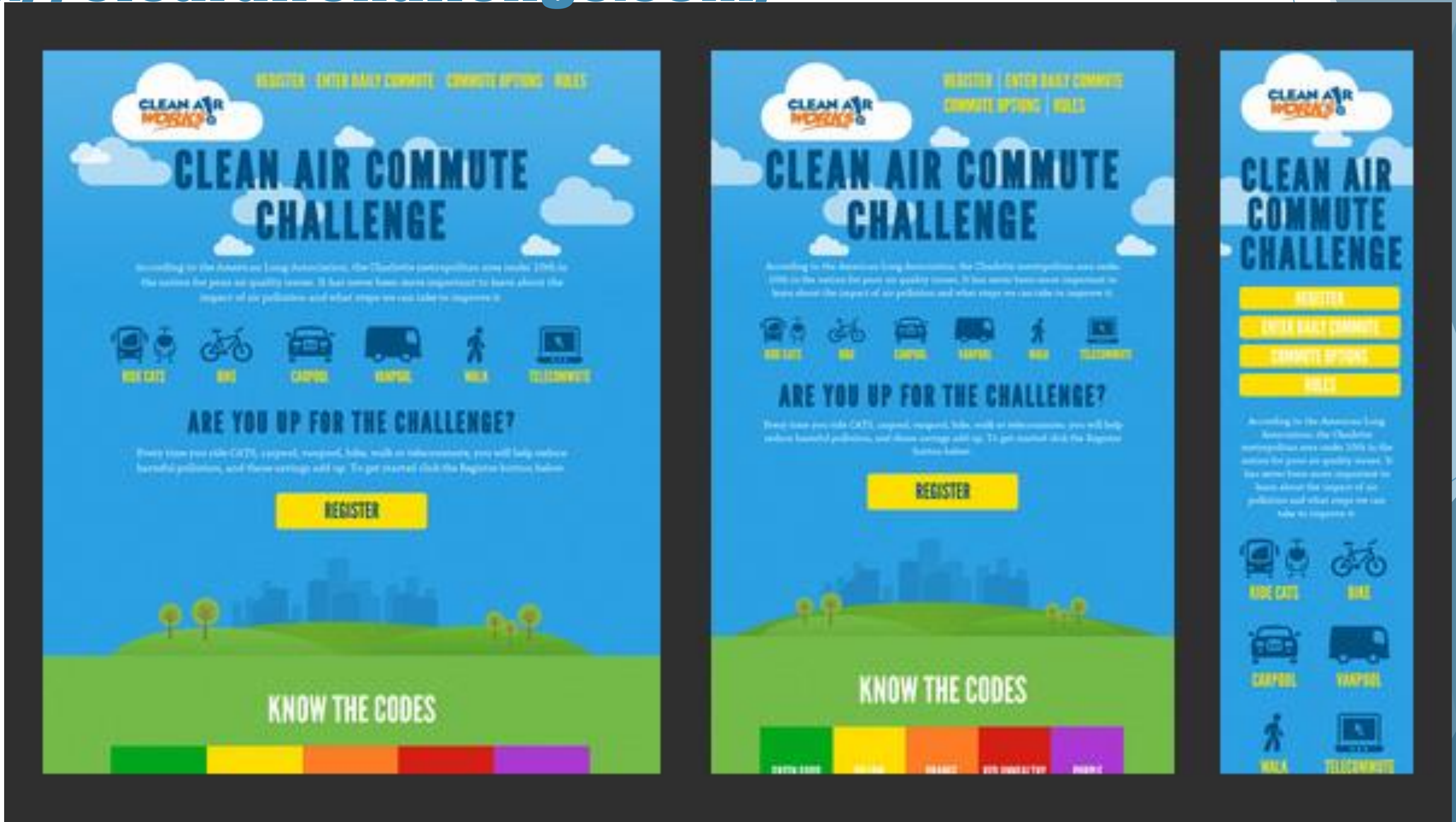
Graphic Design Returns

- Responsive design focuses on the elegant graphic design of the content of a web page regardless of device:
  - It can be difficult for technologist because of the needed aesthetics
  - It can be difficult for designers because of the sophisticated technology
- JavaScript, CSS, and HTML 5 capabilities are leveraged – most notably
  - Fluid layouts
  - Scalable images
  - Media Queries
  - Use of relative sizes (percent/em versus pixels)
- A couple more elegant examples are shown on the next pages

Simon Collison
http://colly.com/



Clean Air Commute Challenge
http://clearairchallenge.com/



Browser Limitations

- Not all browsers are compliant with all features of HTML 5. Some will be left behind.
- The goal is to provide graceful degradation to old browsers.
- The focus is on starting with the capabilities of the most advanced browsers
- A big challenge is a clean design with minimal fixes to accommodate particular browsers and devices
- https://www.caniuse.com

Progressive Enhancement & Graceful Degradation

- Basic content and functionality should be accessible to all browsers
- Enhanced layout is provided by CSS for capable devices
- JavaScript is used as needed to manage browser capabilities
- End user browser control is respected

Graceful degradation

```
<p id="printthis">
  <a href="javascript:window.print()">Print this page</a>
</p>
<noscript>
  <p class="scriptwarning">
    Printing the page requires JavaScript to be enabled.
    Please turn it on in your browser.
  </p>
</noscript>
```

Graceful Degradation

```
<p id="printthis">
  <a href="javascript:window.print()">Print this page</a>
</p>
<noscript>
  <p class="scriptwarning">
    Print a copy of your confirmation.
    Select the "Print" icon in your browser,
    or select "Print" from the "File" menu.
  </p>
</noscript>
```

Progressive enhancement

```
<p id="printthis">Thank you for your order. Please print this page for your records.</p>
<script type="text/javascript">
(function(){
  if(document.getElementById){
    var pt = document.getElementById('printthis');
    if(pt && typeof window.print === 'function'){
      var but = document.createElement('input');
      but.setAttribute('type','button');
      but.setAttribute('value','Print this now');
      but.onclick = function(){
        window.print();
      };
      pt.appendChild(but);
    }
  }
})();
</script>
```

Some specific techniques

- Shift from absolute values to relative values
  - A figure could be 200px by 200px or it could be 50% of the width of the container
  - A font could be 12 pt or 16 px or it could by .8em (which is a percentage of the size of the default document font.)
- Use semantically labeled containers to establish flows
  - Items in a list might float horizontally (float:right;)
  - Objects might shrink or disappear with sizes (Display:none;)

Provide a Structured Structure

- Major page elements should have an id attribute which allows it to be manipulated as a container

```
#content {width: 700px; float: left;}// default for large
@media screen and (max-width: 1024px) {
#content {width: 66%; }}
```

- Elements that are going to be manipulated should have a class element that allows them to be identified

```
@media screen and (max-width: 700px) {
.icon {float: left; margin: .5em 0em .5em 1em; width: 12%;}}
```

Next Steps

- Your instructor will put out an incomplete tutorial example that should help you to get started.
- There are a growing number of tutorials on how to do responsive design. I believe the best initial set (of 15) can be found linked under designwoop.com:
  - http://designwoop.com/2012/03/15-detailed-responsive-web-design-tutorials/

Developing a Responsive Design

- Step 1: Develop a standard layout for a desktop browser
- Step 2: define a style sheet that specifies relative measurements for fonts, images and divs (percent/em versus pixels)
- Step 3: divide the style sheet into styles for:
  - Individual elements – using id – that will control the location and flow of bigger divs
  - Classes of elements – using class – that will control the appearance of selected elements

Developing a Responsive Design

- Step 4: specify a set of media queries that adjust:
  - Containers and how they flow (fluid layouts)
  - Image size or existence
- Step 5: adjust and refine the appearance of the website for the various anticipated formats

Responsive CSS image

- Responsive image

```
img {
   width: 100%;
   height: auto;
}
```

If you want the image to disappear with smaller screen size, use

```
@media screen and (max-width: 480px) {
      img {display: none;
      }
```

The easier way

- BootStrap4 template
  - Lynda (Log in through Pitt)

  https://www.lynda.com/Bootstrap-tutorials/Bootstrap-4-Essential-Training

Demo

- Responsive Design
- Check examples in CourseWeb
- https://www.w3schools.com/bootstrap/bootstrap_get_started.asp
- http://www.pitt.edu/~aaa221/Demo/RespDes2/rddemobasic.htm

For Next week

- Topics
  - DOM
  - Web Scripting
  - Forms
  - AJAX
  - JSON Parsing
- Reading
- Web Programming chapters 8,9,10
- Practice LESS.js
- Assignment 1 due
- Assignment 2 will be issued