CIS 467/602-01: Data Visualization

HTML, CSS, SVG, (& JavaScript)

Dr. David Koop
Assignment 1

• Posted on the course web site
• Due Friday, Feb. 13
• Get started soon!
• Submission information will be posted
• Useful reference: Interactive Data Visualization for the Web by Scott Murray
• myCourses Page
• Registration Issues?
Hyper Text Markup Language (HTML)

- Markup languages allow users to encode the **semantics** of text
- Tags define the boundaries of the structures of the content
  - Tags are enclosed in angle brackets (e.g. `<html>`)
  - Most of the time, you have a start and end tag
  - End tags are just like start tags except that they have forward slash after the open bracket (e.g. `</html>`)  
  - Tags may be nested but not mismatched
    - `<p>A <strong><em>very</em></strong> cool example</p>`
    - `<p>A <strong>very <em>cool</em></strong> example</p>`
  - What about `<img src="mypicture.png" alt="My Image">`?
HTML Elements and Attributes

• Tags denote **elements** of the content (e.g. sections, paragraphs, images)

• Each element may have **attributes** which define other information about the element
  - An attribute has a **key** and **value** (*key*="*value*")
  - e.g. `<img src="mypicture.png" alt="My Image">`

• Many different elements available
  - Common: headers (h1, ..., h6), paragraph (p), lists (ul, ol, li), emphasis (em, strong), link (a), spans & divisions (span, div)
  - Lots more (e.g. abbr): see [MDN Documentation](#)

• Many different attributes available
  - See [MDN Documentation](#): note that some are legacy due to CSS
HTML Element Structure & Naming

• Elements structure a document
  - Document Object Model (DOM)
  - We can visualize this information
  - More importantly, we can navigate this tree

• Identifying and Classifying elements: id and class attributes
  - id identifies a single element—use for a unique case
  - class may identify multiple elements—use for common cases
  - Each element may have multiple classes, separate by spaces
  - Use normal identifiers: don’t start the name with a number
Other HTML Trivia

• `<`, `>`, `&`, and " are special characters, escape with `&lt;`, `&gt;`, `&amp;`, and `&quot;` (note the semi-colon)

• Comments are enclosed by `<!--` and `-->`
  - `<!-- This is a comment -->`

• HTML Documents begin with a DOCTYPE declaration
  - For HTML5, this is easier `<!DOCTYPE html>`

• meta tag: `<meta charset="UTF-8"/>`

• HTML has audio and video tags, math equation support, and more
<!DOCTYPE html>
<html>
  <head>
    <title>A Basic Web Page</title>
  </head>
  <body>
    <h1>My Wicked Awesome Web Page</h1>
    <p><em>This is <strong>cool</strong>. What about <u><strong>this?</strong></u></em></p>
    <img src="patriots.jpg"/>
  </body>
</html>
How to write HTML

• Use a standard text editor
• Use a web site like jsfiddle, jsbox, etc.
• Use an IDE like WebStorm
Cascading Style Sheets (CSS)

• Separate from content, just specifies how to style the content
• Style information can appear in three places:
  - External file
  - In a style element at the beginning of the HTML file
  - In a specific element in the body of a document (least preferable)
• Why Cascading?
  - Don’t want to have to specify everything over and over
  - Often want to use the same characteristics in a region of the DOM
  - Use inheritance: properties that apply to children cascade down
CSS Selectors

• How do we specify what part(s) of the page we want to style?

• The **element types** themselves (the HTML tag)
  - `strong { color: red; }`

• Classes of elements (ties to HTML `class` attribute)
  - `.cool { color: blue; }`

• A **specific** element (ties to HTML `id` attribute)
  - `#main-section { color: green; }`

• Relationships
  - Descendant: `p em { color: yellow; }`
  - Child: `p > em { color: orange; }`

• Pseudo-classes: `a:hover { color: purple; }`
Other CSS Bits

• Comments: /* This is a comment in CSS */
• Grouping Selectors: p, li { font-size: 12pt; }
• Multiple Classes: .cool.temp { color: blue; }
• Colors:
  - Names (Level 1, 2, & 3): red, orange, antiquewhite
  - Dash notation (3- & 6-character): #fff, #00ff00
  - Integer or % RGB and HSL Functions: rgb( 255, 0, 0), rgb(50%, 50%, 0%), hsl(120, 100% 50%)
    - Also background-color
• Watch out for multiple rules (look at how a web browser parses)
• Again, much more documentation at MDN
Sample CSS

body {
    font-face: sans-serif;
    font-size: 12pt;
}

em { color: green; }

em u { color: red; }

em > strong { color: blue; }

img { border: 4px solid red; }

• What colors are displayed for this HTML (with the above stylesheet)?
  - <em>This is <strong>cool</strong>. What about <u><strong>this?</strong></u></em>
How to add CSS to HTML

- External: a separate file via a link element (in the `<head>` section):
  - `<link rel="stylesheet" href="styles.css">`

- Embedded: in the header:
  - `<style type="text/css"> ... </style>`

- Inline: for a specific element: (Discouraged!)
  - `<p style="font-weight: bold;">Some text</p>`
Scalable Vector Graphics (SVG)

• Vector graphics vs. Raster graphics
• Drawing commands versus a grid of pixels
• Why vector graphics?
SVG Background

- Another markup language:
  - Describe the shapes and paths by their endpoints, characteristics
- SVG can be embedded into HTML5 documents!
- Pixel Coordinates: **Top-left** origin

```
(0,0)     (width,0)
```

```
(0,0)    (width,height)
```
SVG Elements

• Drawing primitives:
  - Lines, Circles, Rects, Ellipses, Text, Polylines, Paths
  - Work by specifying information about how to draw the shape
  - Lots more: see MDN Documentation

• Ordering/Stacking:
  - SVG Elements are drawn in the order they are specified
SVG Grouping

• Very powerful, useful for animations and transformations
• `<g> <circle .../> <circle ... /> <circle ... /></g>`

• Can add transforms to the group:
  - `<g transform="translate(0, 50) scale(1, -1)"> ... </g>`

```xml
<svg width="200" height="200">
  <g transform="translate(0, 200) scale(1, -1)">
    <circle cx="50" cy="50" r="10"/>
    <circle cx="80" cy="80" r="10"/>
    <circle cx="110" cy="50" r="10"/>
    <circle cx="140" cy="90" r="10"/>
  </g>
</svg>
```

[SVG Example, Scheidegger, 2015]
SVG Example

<svg id="mysvg" width="400" height="300">
  <circle cx="50" cy="50" r="50"
       style="fill:green; stroke:black; stroke-width:4px"/>
  <rect x="150" y="150" width="50" height="20"
       style="fill:red; stroke: blue; stroke-width: 2px;"/>
  <path d="M 200 10 L 300 10 L 300 50 Z"
        style="fill: none; stroke: red; stroke-width:3px;"/>
</svg>

• Note that the style is separate...
• Paths are raw drawing commands (ever see Logo?)
• What does this look like?
SVG Styles

• Can specify styles via CSS, too

• ...

<svg id="mysvg" width="400" height="300">
  <circle cx="50" cy="50" r="50"/>
  <rect class="normal" x="150" y="150" width="50" height="20"/>
  <path id="p1" d="M 200 10 L 300 10 L 300 50 Z"/>
</svg>

...
JavaScript in one slide

- Interpreted and Dynamically-typed Programming Language
- Statements end with semi-colons, normal blocking
- Variables: `var a = 0;`
- Operators: `+, -, *, /, [ ]`
- Control Statements: `if, else`
- Loops: `for, while, do-while`
- Arrays: `var a = [1,2,3]; a[99] = 100; console.log(a.length);`
- Functions: `function myFunction(a,b) { return a + b; }
- Objects: `var obj; obj.x = 3; obj.y = 5;`
  - Protoypes for instance functions
- Comments are `/* Comment */` or `// Single-line Comment`
JavaScript: Interacting with the DOM

- `window`: global namespace
- `document`
- `document.getElementById(...)"
JavaScript and SVG

• SVG can be navigated just like the DOM
• We can create SVG elements dynamically using JavaScript!
• Example:

```javascript
function addEltToSVG(svg, name, attrs) {
    var element = document.createElementNS(
        "http://www.w3.org/2000/svg", name);
    if (attrs === undefined) attrs = {};
    for (var key in attrs) {
        element.setAttributeNS(null, key, attrs[key]);
    }
    svg.appendChild(element);
}

mysvg = document.getElementById("mysvg");
addEltToSVG(mysvg, "rect", {
    "x": 50, "y": 50,
    "width": 40,"height": 40,
    "fill": "blue"});
```
Tools

• Text Editor
• Chrome DevTools
• MDN Documentation
• Web Environments (jsfiddle, jsbox)
• IDEs (e.g. WebStorm)