Data Visualization (DSC 530/CIS 602-02)

Web Programming

Dr. David Koop
What languages do we use on the Web?
Languages of the Web

- HTML
- CSS
- SVG
- JavaScript
  - Versions of Javascript: ES6, ES2015, ES2017…
  - Specific frameworks: react, jQuery, bootstrap,
HTML Exercise

• What does this HTML mean?

- `<em>This is <strong>cool</strong>. What about <u><strong>this?</strong></u></em>`
CSS Exercise

body {
  font-family: 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>
Interactive Data Visualization by S. Murray

- Free version available on the Web
- http://chimera.labs.oreilly.com/books/1230000000345
- Representing and drawing with data
- Chapter 3 has a nice overview of Web technologies with examples
- Great resource for D3 as well
Office Hours & Email

- Office Hours: Tu 3-5pm, W 2-3pm, F 11am-12pm
- Scheduled office hours are open to all students
- Office hours are first-come, first-serve
- Please be considerate to other students
- No appointment needed to stop in during scheduled office hours
- If you need an appointment outside of those times, please email me with specific details about what you wish to discuss
- Many questions can be answered via email. Do not schedule an appointment to ask a question that could be answered via email
What is the difference between vector and raster graphics?
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

![SVG Diagram]

(0,0)  (width,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

• Paths: directions for drawing
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 Grouping

- Very powerful, useful for animations and transformations
- `<g> <circle .../> <circle ... /> <circle ... /></g>`
- Can add transforms to the group:
  - [http://codepen.io/dakoop/pen/rjpdXp](http://codepen.io/dakoop/pen/rjpdXp)

```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, 2016]
SVG Styles

• Can specify styles via CSS, too

```html
...  
<style type="text/css">
  circle { fill: green; stroke: black;
            stroke-width: 4px; }

  .normal { fill: red; stroke: blue;
            stroke-width: 2px; }

  #p1 { fill: none; stroke: red; stroke-width: 3px; }
</style>
...

<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 with brackets
• Variables: `var a = 0;`
• Operators: `+`, `-`, `*`, `/`, `[ ]`
• Control Statements: `if (<expr>) {...} else {...}`, `switch`
• 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;`
  - Prototypes for instance functions
• Comments are `/* Comment */` or `// Single-line Comment`
JavaScript Objects

- var student = {name: "John Smith", id: "000012345", class: "Senior", hometown: "Fall River, MA, USA"};

- Objects contain multiple values: key-value pairs called **properties**

- Accessing properties via dot-notation: student.name

- May also contain functions:
  - var student =
    
    {firstName: "John",
     lastName: "Smith",
     fullName: function() { return this.firstName + " " + this.lastName; }};

  - student.fullName()

- JavaScript Object Notation (JSON): data interchange format
  - nested objects and arrays (data only, no functions!)
  - **subset** of JavaScript
Objects as Associative Arrays/Dictionaries

• Objects have key-value pairs and can be addressed via those keys, either via dot-notation or via bracket notation: [<key>]

• Example:

```javascript
states = {"AZ": "Arizona", "MA": "Massachusetts", ...};
// Get a state's name given it's abbreviation
console.log("MA is" + states["MA"]);
```

• Similar to dictionaries or associative arrays in other languages (e.g. Python)

• Dot-notation only works with certain identifiers, bracket notation works with more identifiers
Functional Programming in JavaScript

• Functions are first-class objects in JavaScript
• You can pass a function to a method just like you can pass an integer, string, or object
• Instead of writing loops to process data, we can instead use a map/filter/reduce/forEach function on the data that will run our logic for each data item.
  • map: transform each element of an array
  • filter: check each element of an array and keep only ones that pass
  • forEach: run the function for each element of the array
  • reduce: collapse an array to a single object
Function Chaining in JavaScript

• When programming functionally, it is useful to chain functions
• No intermediate variables!
• Often more readable code
• jQuery Example:
  - `$("#myElt").css("color", "blue").height(200).width(320)`
• Used a lot in Web programming, especially D3
• Can return the same object or a new object
• Lazy chaining keeps track of functions to be applied but will apply them later (e.g. when the page loads)
Closures in JavaScript

- Functions can return functions with some values set
- Allows assignment of some of the values
- Closures are functions that "remember their environments" [MDN]
- Example (MDN):

```javascript
function makeAdder(x) {
    return function(y) {
        return x + y;
    };
}
var add5 = makeAdder(5);
var add10 = makeAdder(10);

console.log(add5(2));  // 7
console.log(add10(2)); // 12
```
Manipulating the DOM with JavaScript

- Key global variables:
  - `window`: Global namespace
  - `document`: Current document
  - `document.getElementById(...)`: Get an element via its id

- HTML is parsed into an in-memory document (DOM)
- Can access and **modify** information stored in the DOM
- Can add information to the DOM