Data Visualization (CIS 490/680)

JavaScript

Dr. David Koop
Hyper Text Markup Language (HTML)

• Markup languages allow users to encode the **semantics** of text
• Elements structure a document
  - Elements delineated by tags: `<h1>An element</h1>`
  - Document Object Model (DOM)
  - 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
Cascading Style Sheets (CSS)

- Separate style from content, just specifies how to style the content
- Style information appears in three places: external, head, individual elements
- Statement: `<selectors>: { <style definitions> }
- Cascading:
  - use inheritance idea
  - properties that apply to children cascade down
- Selectors: element types (strong), ids (#main-section), classes (.cool)
  - Can combine to be more specific
    - #main-section em, .cool > strong, p.cool
  - Can group: #main-section, p.cool { font-size: 16pt; }

3
Scalable Vector Graphics (SVG)

- **Vector** graphics vs. Raster graphics
- Another markup language:
  - Describe the shapes and paths by their endpoints, characteristics
  - Lines, Circles, Rects, Ellipses, Text, Polylines, Paths
- SVG can be embedded in HTML5!
- Pixel Coordinates: **Top-left** origin
- We can specify styles of SVG elements in CSS!
Assignment 1

- Link
- HTML, CSS, and SVG
- Due Friday
- Questions?
JavaScript in one slide

• Interpreted and Dynamically-typed Programming Language
• Statements end with semi-colons, normal blocking with brackets
• Variables: \texttt{var a = 0; let b = 2;}
• Operators: $+$, $-$, $\ast$, $\div$, \texttt{[ ]}
• Control Statements: \texttt{if (<expr>) {...} else {...}, switch}
• Loops: \texttt{for, while, do-while}
• Arrays: \texttt{var a = [1,2,3]; a[99] = 100; console.log(a.length);}
• Functions: \texttt{function myFunction(a,b) { return a + b; } }
• Objects: \texttt{var obj; obj.x = 3; obj.y = 5;}
  - Prototypes for instance functions
• Comments are \texttt{/* Comment */} or \texttt{// Single-line Comment}
JavaScript References

- Interactive Data Visualization for the Web, Murray
- MDN Tutorials
JavaScript Objects

- var student = {name: "John Smith", id: "000012345", class: "Senior", hometown: "Peoria, IL, 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", "IL": "Illinois", ...};
// Get a state's name given it's abbreviation
console.log("IL is" + states["IL"]);
```

• 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
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 runs 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
Quiz

• Using `map`, `filter`, `reduce`, and `foreach`, and given this data:
  
  ```javascript
  var a = [6, 2, 6, 10, 7, 18, 0, 17, 20, 6];
  ```

• Questions:
  
  - How would I return a new array with values one less than in a?
  - How would I find only the values >= 10?
  - How would I sum the array?
  - How would I create a reversed version of the array?
Quiz Answers: Notebook

- Data: `var a = [6, 2, 6, 10, 7, 18, 0, 17, 20, 6];`
- How would I subtract one from each item?
  - `a.map(function(d) { return d-1; })`
- How would I find only the values >= 10?
  - `a.filter(function(d) { return d >= 10; })`
- How would I sum the array?
  - `a.reduce(function(s,d) { return s + d; })`
- How would I create a reversed version of the array?
  - `b = [];
    a.forEach(function(d) { b.unshift(d); });`
  - …or `a.reverse() // modifies in place`
- Arrow functions shorten such calls: `a.map(d => d-1); a.filter(d => d >= 10); a.reduce((s,d) => s+d);`
Function Chaining in JavaScript

• When programming functionally, it is useful to chain functions
• No intermediate variables!
• Often more readable code
• jQuery Example:
  
  ```javascript
  $('#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]

```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
```

- **Notebook**
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
Example: JavaScript and the DOM

• Start with no real content, just divs:
  `<div id="firstSection"></div>`
  `<div id="secondSection"></div>`
  `<div id="finalSection"></div>`

• Get existing elements:
  - `document.querySelector`
  - `document.getElementById`

• Programmatically add elements:
  - `document.createElement`
  - `document.createTextNode`
  - `Element.appendChild`
  - `Element.setAttribute`

• Link
Creating SVG figures via JavaScript

- SVG elements can be accessed and modified just like HTML elements
- Create a new SVG programmatically and add it into a page:
  ```javascript
  var divElt = document.getElementById("chart");
  var svg = document.createElementNS("http://www.w3.org/2000/svg", "svg");
  divElt.appendChild(svg);
  
  svg.setAttribute("height", 400);
  svg.setAttribute("width", 600);
  svgCircle.setAttribute("r", 50);
  ```
- You can assign attributes: