Data Visualization (CSCI 627/490)

Data

Dr. David Koop
Scalable Vector Graphics (SVG)

- Vector graphics vs. Raster graphics
- Drawing commands versus a grid of pixels
- Why vector graphics?
JavaScript in one slide

• Interpreted and Dynamically-typed Programming Language
• Statements end with semi-colons, normal blocking with brackets
• Variables: `var a = 0; let b = 2; const c = 4;`
• 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`
Including JavaScript in HTML

• Use the script tag
• Can either inline JavaScript or load it from an external file

    - `<script type="text/javascript">
      a = 5, b = 8;
      c = a * b + b - a;
    </script>
    <script type="text/javascript" src="script.js"/>

• Script tag can reference local or **remote** external javascript files
• The order the javascript is in is the order it is executed
• Example: in the above, `script.js` can access the variables `a`, `b`, and `c`
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

• Always works via bracket-notation: student["name"]

• May also contain functions:
  - var student = {firstName: "John", lastName: "Smith", fullName: function() { return this.firstName + " " + this.lastName; }};
  - student.fullName()
Assignment 1

• Due Tonight
• Write HTML, CSS, and SVG
• Use Plot library
• Text markup and styling (information)
• Drawing markup and styling (tower)
• Draw Bar chart
• Questions?
Assignment 1

• Due Tonight
• Write HTML, CSS, and SVG
• Use Plot library
• Text markup and styling (information)
• Drawing markup and styling (tower)
• Draw Bar chart
• Questions?
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:
  - `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:
  - `$('#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
Example: JavaScript and the DOM

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

- Get existing elements:
  - `document.querySelector`/`querySelectorAll`
  - `document.getElementById`

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

---

Bears

Chicago, IL

2018-2019 NFC North Champions

What will happen this year?
Observable's HTML Templating

- Allows JavaScript expressions to be **inlined** in HTML (or SVG content)
- Use `{{...}}`
- Example:
  - [JavaScript] `name = "Prof. Koop"
  - [HTML] `<p>Hello, my name is ${name}</p>`
Using Observable's HTML Templating

```html
<div id="firstSection">
  <h1>Bears</h1>
  <p>Chicago, IL</p>
</div>
<div id="secondSection">
  <h2>2018-2019 NFC North Champions</h2>
</div>
<div id="finalSection">
  ${scores.map((game) => html`<p>${game.date}:
    ${game.win ? "Win" : "Loss"} (${game.score})</p>`)}
  <img src="...Justin_Fields....jpg" width="240"></img>
  <p>What will happen this year?</p>
</div>
```

Notebook
SVG Manipulation Example

- Draw a horizontal bar chart
  ```javascript
  var a = [6, 2, 6, 10, 7, 18, 0, 17, 20, 6];
  ```

- Steps?

D. Koop, CSCI 627/490, Fall 2022
SVG Manipulation Example

• Draw a horizontal bar chart
  - \texttt{var a = [6, 2, 6, 10, 7, 18, 0, 17, 20, 6];}
• Steps:
  - Programmatically create SVG
  - Create individual rectangle for each item
• Notebook
...or Use Templating

- Same with SVG as with HTML
- Notebook
“Computer-based visualization systems provide visual representations of datasets designed to help people carry out tasks more effectively.”

— T. Munzner
Data

•What is this data?

<table>
<thead>
<tr>
<th>R011</th>
<th>42ND STREET &amp; 8TH AVENUE</th>
<th>00228985</th>
<th>00008471</th>
<th>00000441</th>
<th>00001455</th>
<th>00000134</th>
<th>00033341</th>
<th>00071255</th>
</tr>
</thead>
<tbody>
<tr>
<td>R170</td>
<td>14TH STREET-UNION SQUARE</td>
<td>00224603</td>
<td>00011051</td>
<td>00000827</td>
<td>00003026</td>
<td>00000660</td>
<td>00089367</td>
<td>00199841</td>
</tr>
<tr>
<td>R046</td>
<td>42ND STREET &amp; GRAND CENTRAL</td>
<td>00207758</td>
<td>00007908</td>
<td>00000323</td>
<td>00001183</td>
<td>00003001</td>
<td>00040759</td>
<td>00096613</td>
</tr>
</tbody>
</table>

•Semantics: real-world meaning of the data
•Type: structural or mathematical interpretation
•Both often require metadata
  - Sometimes we can infer some of this information
  - Line between data and metadata isn’t always clear
Semantics

- The meaning of the data
- Example: 94023, 90210, 02747, 60115
Semantics

- The meaning of the data
- Example: 94023, 90210, 02747, 60115
  - Attendance at college football games?
Semantics

• The meaning of the data
• Example: 94023, 90210, 02747, 60115
  - Attendance at college football games?
  - Salaries?
Semantics

• The meaning of the data
• Example: 94023, 90210, 02747, 60115
  - Attendance at college football games?
  - Salaries?
  - Zip codes?
• Cannot always infer based on what the data looks like
• Often require semantics to better understand data
• Column names help with semantics
• May also include rules about data: a zip code is part of an address that uniquely identifies a residence
• Useful for asking good questions about the data