CSC 544 Data Visualization

Joshua Levine
josh@arizona.edu

Lecture 03
d3 Intro

Jan. 23, 2023

Today's Agenda

- Reminders:
  - A00 due
  - A01 posted
- Goals for today:
  - Wrap up Javascript introduction
  - Discuss how Javascript can be used to manipulate the DOM
  - And then introduce d3.js

A Few More Tips in Javascript

Important In-Class Activity We Didn’t Get To in L02 (from https://cscheid.net/courses/fall-2019/csc444/lectures/lecture3/activities.html)

- Write a procedure `map` that takes two parameters: an array `lst` and another procedure `f`. The procedure you’ll write should iterate over the array and return a new array with the result of applying `f` to every object.

Important In-Class Activity We Didn't Get To in L02 (from https://cscheid.net/courses/fall-2019/csc444/lectures/lecture3/activities.html)

- Write a procedure `map` that takes two parameters: an array `lst` and another procedure `f`. The procedure you'll write should iterate over the array and return a new array with the result of applying `f` to every object.

- Answer:

```
function map(lst, f) {
  let result = [];
  for(let i=0; i<lst.length; i++) {
    result.push(f(lst[i]));
  }
  return result;
}
```

Built-In Javascript Enumerations

- Starting with: `let array = [1,2,3,4,5];`

- `forEach` (call a function for each element):
  ```
  array.forEach((x,i,a) => console.log(x,i,a[i]));
  ```

- `map` (create an array of function outputs):
  ```
  let map1 = array.map(x => x * 2);
  ```

- `filter` (use function to sub select elements of array):
  ```
  let result = array.filter(x => x > 3);
  ```

- Also will see `reduce()`, and many other similar helper functions

Enumerable Objects

- If one wants to iterate over the keys in an object:

```
for (let key in obj) {
  obj[key] // access value for key
}

keys(obj); //returns a list of keys
```

- Compare with iterating over elements in an array:

```
for (let i=0; i<array.length; i++) {
  obj[i] // access value for key
}
```

Accessing the DOM

Data Structures Available in the Browser

- In addition to the `window`, also have access to the `document` object in Javascript
- The `window` is the literally the “window” for which the current script is running
    - Can be used to force the window the refresh (as we’ll see), accessing variables, as well as certain browser specific calls.
- The `document` is the currently loaded HTML document, organized as a DOM

Accessing DOM Nodes

- In Javascript, the `document` variable has full access to the DOM itself
- One can query the document to find specific nodes:
    - For elements with ids use `document.getElementById()`
    - `document.querySelector()` and `document.querySelectorAll()` use CSS-like selectors
    - `document.getElementsByTagName()` and `document.getElementsByClassName()` return matching lists

Manipulating the DOM

- Can ask the document for new elements:
  ```
  newnode = document.createElement(“sometag”);
  ```
- Given an element, can add to the tree:
  ```
  node = document.getElementById(“nodeid”);
  node.appendChild(newnode);
  ```
- Can also create text nodes:
  ```
  text = document.createTextNode(“my text”);
  node.appendChild(text);
  ```

Manipulating DOM Elements

- Given an element, one can also manipulate its attributes:

```
node = document.getElementById(“nodeid”);

node.setAttribute(“style”,
      “background-color: black;”);
```

- Alternatively:

```
node.style.backgroundColor = “blue”;
```

- In addition to standard html/css attributes, we will also see situations where we attach new fields to DOM elements

Break for Questions/Demo of SVG+Javascript

https://cscheid.net/courses/fall-2019/csc444/lectures/lecture4.html

d3js.org

Overview Examples Documentation API Source

Data-Driven Documents

Like visualization and creative coding? Try interactive JavaScript notebooks in **Observable!**

D3.js is a JavaScript library for manipulating documents based on data. **D3** helps you bring data to life using HTML, SVG, and CSS. D3's emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization

See more examples.

Selections

Selecting Elements w/ d3

- `d3.select()` and `d3.selectAll()` both accept a CSS selector and return elements
    - Replaces `document.getElementById()`, `document.querySelector()`, etc.
- `.append()` can then be used to insert elements in the DOM at the current selection
- `.text()` can be used to insert text between tags

Setting Attributes w/ Anonymous Functions

- Given a d3 selection
  - `.attr(attr, value)` can be used to set attributes
  - `.style(attr, value)` can be used to set CSS styles
- Both accept anonymous functions, e.g.,

  -
    ```
    .style(“width”, function() {
        return Math.random() * 100;
    });
    ```

    This would set the width of the selection to a random value between 0 and 100.

Data Joins

Binding Data

- Given a selection in d3, once can bind data to it using `.data()`
- This builds a mapping between each element in the selection and each data element
  - One can control this in lots of ways, but the default is sequential, element `i` is mapped to data at index `i`.

Accessing Bound Data

- Once bound, one can use the data to define attributes:

  -
    ```
    .style("width", function(d) {
        return d * 100;
    });
    ```

    This would set the width of each element in the selection to d*100.

  - Can also use `function(d,i)` if one wants to access the index `i` of the data element in addition to its value `d`

Lec04 Reading

- Lecture notes on scales in d3.js:
    - https://cscheid.net/courses/fall-2019/csc444/lectures/lecture6.html
- Murray, Chapter 7
- See also (recommended)
    - d3.js drills: https://cscheid.net/projects/d3-drills/

Assignment 01

Assigned: Monday, January 23
Due: Monday, February 6, 4:59:59 pm