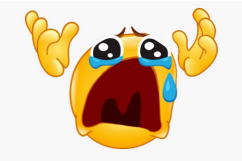
Lab 3: Introduction to D3

CS 7450: Information Visualization Spring 2026

Georgia Tech

Reminders

- Please put your teams on the Prototype Team Sign Up Excel Sheet!
- Homework 1 is due Friday, February 6th at 11:59PM. Good luck!!!
- Lab 1 and Lab 2 supplemental notes are live on the Lab 1 and Lab 2 assignments. Check them out!!!



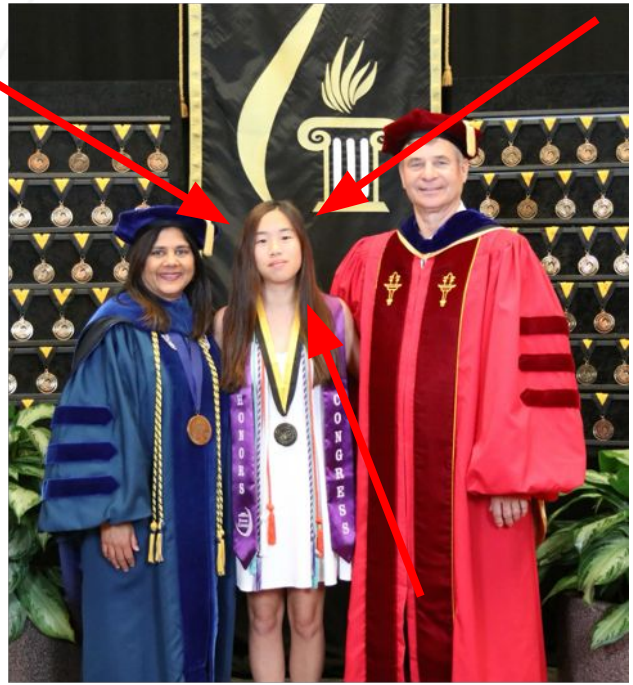
Literally me because I don't know if you are reading the supplemental notes and stuff

Georgia Tech

Lab 3 Instructions

Georgia Tech

In case you forgot



Alysha

Jack

Georgia Tech

Goals for the labs

- ~~Learn web fundamentals~~
  - ~~HTML, CSS, SVG, JS, DOM, and various tooling methods~~
- D3 Fundamentals
  - Data loading, binding, scalings, axes, and various charts
- D3 Interactions
  - Event listeners, selection, brushing and linking, sorting, and transitions
- Advanced D3 techniques
  - Network visualizations, scrollytelling, and geospatial projections

Oh my gosh. It’s time for some D3

What is D3?

- D3.js (also known as D3 or **Data-Driven Documents**) is a free, open-source JavaScript library for creating dynamic data visualizations on the web
- Created by Mike Bostock, Jeffrey Heer, and Vadim Ogievetsky in 2011 at Stanford University's Stanford Visualization Group
- Designed to replace other such framework such as Protovis, Prefuse, and Flare by creating a more expressive framework and improved performance on web applications.

Quick Tangent before we get more into D3

Anonymous Functions

- An **anonymous function** is a function that does not have a standard function declaration
  - Basically a no-name function
- Typically these functions are used inline as a value, most often as a callback in JavaScript
- These will be incredibly useful as we continue talking about D3 (I am going to show these a lot!)

```
const double = function(x) { return x * 2; };
```

Anonymous function expression

```
const double = (x) => x * 2;
```

Anonymous arrow function

```
d3.selectAll("circle")
  .data(data)
  .attr("r", d => d.value);
```

Example of an anonymous function in a D3 method chain

Georgia Tech

Okay back to D3

Jack's Principles of D3

- While there really doesn't exist a set of rigid D3 principles, here are some **principles** that I feel describe D3 as a framework
- **Principles**
  - **Selection**
  - **Data Binding**
  - **Modification**
  - **Enter-Update Exit Pattern**
  - **Object Oriented Programming in D3** ← (we will probably cover this in a future lab as it is more related to the creation of customized visualizations)
- We will be using these principles throughout the rest of the course so please get comfortable with them

Selection

- D3 operates by **selecting** and manipulating DOM objects dynamically through the use of JavaScript code
  - Selection is generally the first thing that you will do when manipulating the DOM
- Selections are also **immutable** and **chainable**
  - Each method call returns a new selection, enabling method chaining

```
d3.select("something");
d3.selectAll("something");
```

Example of selection

```
d3.select("svg")
  .selectAll("circle")
  .attr("fill", "steelblue");
```

Example of selecting all circles and coloring them blue

Georgia Tech

Data Binding

- In D3 we have the ability to **bind data** to DOM elements
  - This establishes a relationship between each element in the selection and each item in the data array
- **Key functions** are also valuable here as we can specify each element on a unique identifier rather than position within the data
  - Useful for animating transitions, reordering data, and dynamic data sets
- **After binding occurs**, data is automatically passed into functional accessor

```
const data = [10, 20, 30]; // Data

d3.select("svg")
  .selectAll("circle")
  .data(data)
  .join("circle")
  .attr("cx", (d, i) => i * 40 + 20) // Function with d (data) and i (index)
  .attr("cy", 50)
  .attr("r", d => d);   // Data
```

Example of data binding

```
const data = [
  { id: "a", value: 10 },
  { id: "b", value: 20 },
  { id: "c", value: 30 }
];

d3.select("svg")
  .selectAll("circle")
  .data(data, d => d.id)   // Key function
  .join("circle")
  .attr("cx", (d, i) => i * 50 + 25)
  .attr("cy", 50)
  .attr("r", d => d.value);
```

Example of key function

Georgia Tech

Modification

- In D3 we have the ability to **modify** attributes, styles, HTML tags, and text of DOM elements
- This updates the visual representation of the selection to accurately reflect the values in the data
- **Accessor functions** are also valuable here as we can modify property values dynamically based on the data rather than static constants

```
d3.selectAll("circle")
  .style("fill", "steelblue")
  .style("stroke", "black");
```

Example of modifying style

```
d3.selectAll("circle")
  .attr("r", 15);
```

Example of modifying an attribute

Enter-Update Exit Pattern

- In D3 we have the ability to manage the lifecycle of elements as the data changes
- D3 separates this management into three different states being: **enter**, **update**, and **exit**
  - **Enter -** new data with no element - **create elements**
  - **Update -** existing elements are matched to data - **modify elements**
  - **Exit -** elements with no data - **remove elements**

```
const circles = d3.select("svg")
  .selectAll("circle")
  .data(data);

// ENTER
circles.enter()
  .append("circle")
  .attr("r", 5);

// UPDATE
circles
  .attr("cx", d => d.x)
  .attr("cy", d => d.y);

// EXIT
circles.exit().remove();

// YAY!!! WE ARE DONE
```

Example of a classic enter-update exit pattern

```
// ENTER UPDATE EXIT ALL IN ONE!!!
d3.select("svg")
  .selectAll("circle")
  .data(data)
  .join("circle")
  .attr("cx", d => d.x)
  .attr("cy", d => d.y)
  .attr("r", 5);
```

Example of join method

Georgia Tech