Module 12: D3.js - Day 3 Lesson Notes

Lesson 1: Intro to D3.js

Introduction to D3

D3, or Data-Driven Documents, is a JavaScript library that is primarily used for creating dynamic and interactive data visualizations in the web browser. It makes use of the widely implemented SVG, HTML5, and CSS standards. It was developed by Mike Bostock with the goal of providing a tool for efficient data manipulation and visualization.

Overview of D3.js

D3.js allows you to bind arbitrary data to a Document Object Model (DOM), and then apply data-driven transformations to the document. For example, you can use D3.js to generate an HTML table from an array of numbers. Or, use the same data to create an interactive SVG bar chart with smooth transitions and interaction.

Benefits and Uses of D3

D3.js is incredibly flexible; it allows a high level of customization and is not limited to a pre-defined set of charts. Its data-driven approach allows for efficient updating and modification of elements. D3.js supports large datasets and dynamic behaviors for interaction and animation. D3's functional style allows code reuse through a diverse collection of official and community-developed modules.

Careers Using D3

D3.js is often used by data scientists, data analysts, and web developers who want to create impressive data visualizations. Knowing D3.js can open career opportunities as a data visualization developer, data analyst, data scientist, business intelligence analyst, or frontend web developer.

Setting Up D3

To get started with D3, you first need to include the library in your HTML file. This is often done by adding a <script> tag in the HTML file that points to the D3.js library hosted on a CDN:

```
<script src="https://d3js.org/d3.v5.min.js"></script>
```

Creating an Element with D3

Let's start with a simple example of creating an SVG rectangle using D3. Assume you have an SVG element already added to your HTML:

```
<svg width="500" height="500" id="svg"></svg>
```

You can use D3 to add a rectangle to the SVG as follows:

```
d3.select("#svg")

  .append("rect")

  .attr("width", 50)

  .attr("height", 100)

  .attr("fill", "blue");
```

This code selects the SVG element with the id "svg", appends a "rect" element (SVG for rectangle), and sets the width, height, and fill color of the rectangle.

Creating a Simple Bar Chart with D3

Creating a simple bar chart involves binding a dataset to rectangles (for the bars) and setting the width, height, and position of each rectangle based on the data. Here's a basic example:

```
var data = [80, 120, 60, 150, 200];

var svg = d3.select("#svg");


svg.selectAll("rect")

  .data(data)

  .enter()

  .append("rect")

  .attr("x", (d, i) => i * 50)

  .attr("y", d => 200 - d)

  .attr("width", 40)
```

```
  .attr("height", d => d)

  .attr("fill", "orange");
```

This code selects all "rect" elements in the SVG, binds the data array to them, creates a new rectangle for each data item, and sets the "x", "y", "width", "height", and "fill" attributes for each rectangle. The "x" and "y" attributes set the position of the rectangle, and the "height" is determined by the data value.

Lesson 2: D3 Deep Dive Part 1

SVG Elements

Scalable Vector Graphics (SVG) is an XML-based vector image format for two-dimensional graphics. In D3, you can generate and manipulate SVG elements to create visualizations. For example, you can create a circle with D3 and SVG like so:

```
d3.select('svg')

  .append('circle')

  .attr('cx', 50)

  .attr('cy', 50)

  .attr('r', 40)

  .attr('fill', 'red');
```

Bar Charts

We've already touched on this in the previous lesson. Here, you bind data to rectangles (the bars) and manipulate their height based on the data:

```
var data = [80, 120, 60, 150, 200];

var svg = d3.select("#svg");


svg.selectAll("rect")

  .data(data)
```

```
  .enter()
  .append("rect")
  .attr("x", (d, i) => i * 50)
  .attr("y", d => 200 - d)
  .attr("width", 40)
  .attr("height", d => d)
  .attr("fill", "orange");
```

Line Charts

In D3, you can use the d3.line() generator to create a line chart. You need to pass an array of [x, y] pairs to the line generator. A basic example:

```
var data = [[0, 50], [1, 200], [2, 100], [3, 170], [4, 150], [5, 200]];
var line = d3.line();

svg.append("path")
  .attr("d", line(data))
  .attr("stroke", "blue")
  .attr("fill", "none");
```

Scatter Plots

A scatter plot in D3 can be created by mapping data to the positions of circles:

```
var data = [[30, 200], [85, 100], [200, 210], [320, 400], [400, 50]];
svg.selectAll("circle")
  .data(data)
  .enter()
  .append("circle")
  .attr("cx", d => d[0])
```

```
  .attr("cy", d => d[1])
  .attr("r", 10);
```

Loading Data

D3 provides methods for loading data from different formats including CSV, JSON, and TSV. For instance, to load a CSV file:

```
d3.csv("/path/to/data.csv").then(function(data) {
  console.log(data);
});
```

Binding Data to Visual Elements

To bind data to visual elements, you use the data() function in D3. This function is used to join the specified array of data to the selected elements:

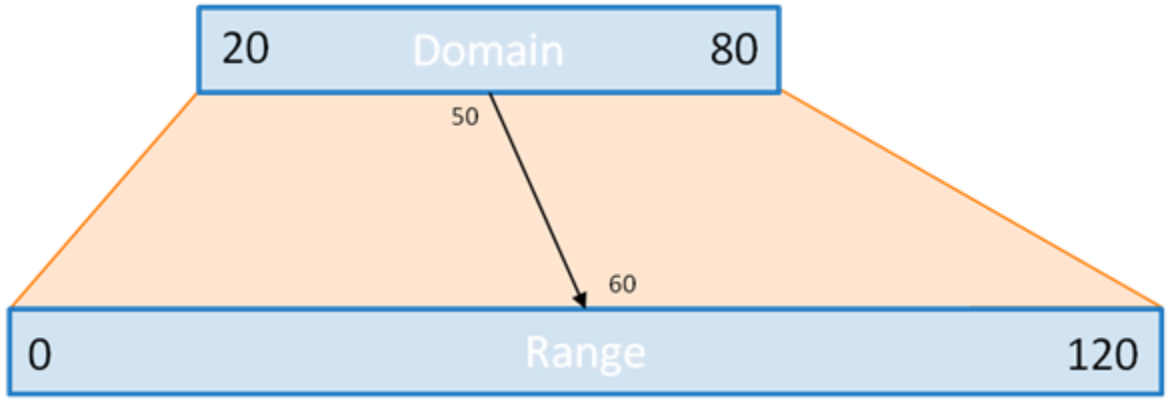
```
var data = [80, 120, 60, 150, 200];
var circles = svg.selectAll("circle")
  .data(data);
```

The data() function is often used with enter(), exit() and join() for handling dynamic data:

```
var data = [80, 120, 60, 150, 200];
svg.selectAll("circle")
  .data(data)
  .join(
      enter => enter.append("circle").attr("r", d => d),
      update => update.attr("r", d => d),
      exit => exit.remove()
  );
```

The enter() function creates new elements for data not yet bound to visual elements. The update section modifies existing elements to match current data. The exit() function removes elements for which there is no longer any corresponding data.

Lesson 3: D3 Deep Dive - Part 2



Scales

Scales are a key component of D3 that allow you to map data values to visual variables. The simplest type is the linear scale, which maps a continuous domain to a continuous range. Here's a simple linear scale that maps the domain [0, 100] to the range [0, 200]:

```
var scale = d3.scaleLinear()
  .domain([0, 100])
  .range([0, 200]);

console.log(scale(50));  // Outputs 100
```

Axes

Axes visualize scale mappings. They can be created using D3's axis functions:

```
var scale = d3.scaleLinear()
  .domain([0, 100])
  .range([0, 200]);

var axis = d3.axisBottom(scale);
```

```
svg.append("g")
  .call(axis);
```

Labels

Labels can be created by appending text elements to SVG:

```
svg.append("text")
  .attr("x", 100)
  .attr("y", 50)
  .text("This is a label");
```

Interactivity and Animations Part 1: Mouse Events

D3 allows you to add event listeners to SVG elements for user interaction:

```
svg.selectAll("circle")
  .on("mouseover", function() {
      d3.select(this)
      .attr("fill", "red");
  });
```

Interactivity and Animations Part 2: Transitions

Transitions can be used to animate changes in D3 visualizations:

```
svg.selectAll("circle")
  .transition()
  .duration(2000)
  .attr("r", 100);
```

Easing Functions

D3 transitions support easing functions, which can be used to control the pacing of the animation:

```
svg.selectAll("circle")

  .transition()

  .duration(2000)

  .ease(d3.easeBounce)

  .attr("r", 100);
```

Data Manipulation Part 1: Filtering Data

D3 arrays have a filter function that you can use to filter data:

```
var data = [80, 120, 60, 150, 200];

var filteredData = data.filter(function(d) { return d > 100; });
```

Data Manipulation Part 2: Sorting Data

D3 arrays also have a sort function for sorting data:

```
var data = [80, 120, 60, 150, 200];

var sortedData = data.sort(d3.ascending);
```

Layouts: Creating a Pie Chart

D3 provides layout functions that can help generate data for complex visualizations. Here's how you could use the pie layout to create a pie chart:

```
var data = [80, 120, 60, 150, 200];

var pie = d3.pie();



var arcData = pie(data);



svg.selectAll("path")
```

```
  .data(arcData)

  .enter()

  .append("path")

  .attr("d", d3.arc().innerRadius(0).outerRadius(100))

  .attr("fill", "orange");
```

Resources for Learning D3.js

D3.js Official Documentation: This is the most comprehensive resource for D3.js. It covers all the functionalities provided by D3.js.

Observable's Introduction to D3: This interactive tutorial by Observable covers the basics of D3.js.

FreeCodeCamp's Data Visualization with D3: A free online course that covers D3.js as part of data visualization. From the course: https://www.freecodecamp.org/learn/data-visualization/

D3.js Tutorial – Data Visualization for Beginners

The D3.js Graph Gallery

Additional Tutorials: https://website.education.wisc.edu/~swu28/d3t/concept.html & https://www.tutorialsteacher.com/d3js/create-bar-chart-using-d3js

Vocabulary Terms and Definitions

Selections: The core concept in D3.js, used to grab elements from the page to manipulate them.

Data Join: The process of tying data to our selection.

Enter and Exit: D3.js concepts used to handle the entering and exiting of elements when data changes.

Scales: Functions that map from an input domain to an output range, useful for manipulating visual dimensions based on data.

Axis: D3.js provides functions to automatically generate axes for your charts.

SVG (Scalable Vector Graphics): An XML-based vector image format for two-dimensional graphics, used by D3.js to create visualizations.

Transitions: Used to animate changes in D3.js visualizations.

D3 Layouts: Pre-defined functions that help in organizing elements based on your data.

Cheat Sheet for D3.js Calculations

x and y coordinates: In D3.js, you often set the 'x' and 'y' coordinates of SVG elements to position them. Typically, 'x' and 'y' are set using scales, which are functions that map from an input domain (your data) to an output range (pixels on the screen). For example, when creating a bar chart, you might use a band scale to set 'x' (to position each bar along the x-axis) and a linear scale to set 'y' (to position each bar vertically based on its value).

Width and Height: The width and height of SVG elements are also often set using scales. For a bar chart, you might use the bandwidth of the x-scale to set the width of each bar, and use the y-scale to set the height of each bar (which would be the chart height minus the 'y' position of the bar).

Domain and Range: In D3.js scales, the domain is the range of input values for your data, and the range is the range of output values, usually in pixels. For example, for a y-scale in a bar chart, the domain might be [0, max value of your data] and the range might be [height of your chart, 0].

d3.max and d3.min: These are D3.js functions used to find the maximum and minimum values in your data, often used to set the domain of scales.

Margins: In D3.js charts, margins are used to provide space around your chart area for things like axes labels and titles. The width and height of the chart area are typically calculated as the total width/height minus the margins.

Transforms: In D3.js, transforms are used to shift SVG elements (like groups of bars or axes) by a certain amount. For example, a left margin is implemented by translating the chart area to the right by the margin amount.

Codepen Link

https://codepen.io/shafferma08/pen/qBQJNvM