Learn D3.js Textbook Notes

[1. Introduction 2](#_Toc169103450)

[a. D3 data driven documents 2](#_Toc169103451)

[b. What is D3? 2](#_Toc169103452)

[c. How does it work? 2](#_Toc169103453)

[d. Using D3 2](#_Toc169103454)

[e. Hello World 2](#_Toc169103455)

[f. Debugging D3 2](#_Toc169103456)

[g. Modules (microlibraries) 3](#_Toc169103457)

[2. Technical Fundamentals 3](#_Toc169103458)

[a. SVG – Scalable Vector Graphics 3](#_Toc169103459)

[b. Viewport – SVG graphics context 3](#_Toc169103460)

[c. Shapes 3](#_Toc169103461)

[d. Fills and Strokes 3](#_Toc169103462)

[e. SVG styling with CSS 3](#_Toc169103463)

[f. Transparency 3](#_Toc169103464)

[g. Text 3](#_Toc169103465)

[h. Group Containers <g> 3](#_Toc169103466)

[i. Reusing Object: use and defs 3](#_Toc169103467)

[j. Matrix Transforms 3](#_Toc169103468)

[k. Configuring the SVG viewport 4](#_Toc169103469)

[l. Gradients 4](#_Toc169103470)

[m. Clipping 4](#_Toc169103471)

[n. Filters 4](#_Toc169103472)

[o. Essential JS data structures 4](#_Toc169103473)

[p. Data Formats 4](#_Toc169103474)

[3. Quick Start 4](#_Toc169103475)

[a. Selecting and Binding Data 4](#_Toc169103476)

[b. Creating a Bar Chart with D3 and HTML 5](#_Toc169103477)

[c. Creating a Bar chart with D3 and SVG 5](#_Toc169103478)

[d. Updating Data Visualizations 5](#_Toc169103479)

[4. Data Binding 5](#_Toc169103480)

[a. Selections 5](#_Toc169103481)

[b. D3 general update pattern 6](#_Toc169103482)

[c. Loading and Parsing Data 6](#_Toc169103483)

[d. Complete step by step example – join() 6](#_Toc169103484)

[5. Manipulating Data and Formatting 6](#_Toc169103485)

[a. Manipulating arrays collections and objects 6](#_Toc169103486)

[b. Random Numbers 6](#_Toc169103487)

[c. Interpolation 6](#_Toc169103488)

[d. Number Formatting 6](#_Toc169103489)

[e. Date and time parsing and formatting 6](#_Toc169103490)

[6. Scales Axes and Colors 6](#_Toc169103491)

[a. Axes 6](#_Toc169103492)

[b. Scales 7](#_Toc169103493)

[c. Color Palettes, schemes, and spaces 9](#_Toc169103494)

# Introduction

## D3 data driven documents

* + 1. Open source JS library
    2. Used to create interactive web based data visualizations

## What is D3?

* + 1. D3 focuses on the data but also use HTML CSS and SVG

## How does it work?

* + 1. Binds data and updates svg base don adding removing or changing the data
    2. Use CSS selectors to select one or more elements which can be used to apply styling and change the attributes of selected elements

## Using D3

* + 1. bl.ocks.org
    2. observablehq.com

## Hello World

## Debugging D3

* + 1. JS console

## Modules (microlibraries)

* + 1. Don’t always have to load the entire D3 library

# Technical Fundamentals

## SVG – Scalable Vector Graphics

* + 1. Xml based image format that describes graphics using geometrical attributes
    2. Object based and provides a DOM

## Viewport – SVG graphics context

* + 1. When SVG is embedded in HTML it creates a viewport
    2. Default graphics context similar to the context created by Canvas
    3. Adding svg tag creates the viewport
    4. Can change orientation scale and other aspects of coordinate system by configuring the *viewBox* attribute

## Shapes

## Fills and Strokes

* + 1. Shapes have default black fill colors and transparent stroke borders

## SVG styling with CSS

* + 1. Many CSS properties only exist in SVG
    2. HTML – background-color and color
    3. SVG – fill and stroke
    4. Elements inside SVG are SVG namespace but SVG itself in an HTML uses HTML namespace

## Transparency

* + 1. CSS zindex doesn’t work in SVG
    2. To move an object to the front you have to modify the DOM tree (selection.raise())

## Text

* + 1. Y is the baseline – need to account for height or make alignment-baseline=”middle”
    2. Text-anchor => start is left end is right
    3. If text spans multiples lines then use tspan inside text to move words or letters to positions relative to parents

## Group Containers <g>

* + 1. Invisible element and its positioned at the center of coordinates

## Reusing Object: use and defs

* + 1. Defs
       1. Elements that will not be displayed unless called by id
    2. Use
       1. References an existing elemnt by ID using standard xlink notation
    3. Good for clipping masks, filters, and gradients

## Matrix Transforms

* + 1. ORDER MATTERS
    2. Flipping an object can be achieved by scaling with neg values for x and or y

## Configuring the SVG viewport

* + 1. Changing the values in the viewbox center cords width height
    2. Rarely use viewbox just use transforms

## Gradients

* + 1. Stop offset = # 0-1 stop-color=rgb

## Clipping

* + 1. clipPath = “url(#id)”

## Filters

## Essential JS data structures

* + 1. **ES6**
       1. Use *const* and let not *var*
       2. Arrow functions *d => d* instead of *function(d){return d;}* when appropriate
       3. Spread operators [… iterable] chained expressions, maps, sets and promises
       4. Template string literals, defined using backticks
       5. Iterable collections maps and sets
    2. **Arrays**
       1. For
       2. forEach()
       3. map
       4. reduce
    3. **Strings**
       1. Multiline
          1. Add a backslash \ at the end of each line
       2. Template Literals
          1. `` - allow inclusion of js expressions
       3. Special characters in string
          1. Precede it with a backslash
       4. All methods return new string
    4. **Functions**
    5. **Objects**
       1. {}
       2. For (let key in color){ log(key + “:” + color[key]) }
    6. **Maps and sets**
       1. Sets – don’t allow repeated values
       2. Map – key -value pairs

## Data Formats

* + 1. CSV, XML, JSON

# Quick Start

## Selecting and Binding Data

* + 1. **Selecting and Appending**
       1. Returns a d3 handle for a node or a set of nodes
       2. Can convert to d3 by calling node() or nodes()
       3. Select() – will always return only one object, if selector matches more tiems only first will be returned
       4. SelectAll() – returns a collection can be iterated with each()
    2. **Binding Data**
       1. Data, datum
          1. Receives an array that maps to DOM elements
       2. Join, enter, exit
          1. Binds the data to DOM elements

## Creating a Bar Chart with D3 and HTML

* + 1. Binding data to HTML
    2. **Scales**
       1. Domain – dimension of data
       2. Range – dimension of graphics context displayed
    3. **Array utilities**
       1. D3.max
       2. D3.sort
       3. D3.ascending
       4. D3.descending
    4. **Adding Labels**

## Creating a Bar chart with D3 and SVG

## Updating Data Visualizations

* + 1. **Handling Events**
       1. On()
          1. String, handler function

# Data Binding

## Selections

* + 1. **W3C Selectors**
       1. \* - all elements
       2. [] – for attribute selectors
       3. . – for class selectors
       4. # - for id selectors
       5. : - for pseudo classes, elements base don characteristics such as state properties position and negation
       6. Combinators – contextual selections of existing nodes
          1. Descendants
          2. Child
          3. Next-sibling
          4. Subsequent-sibling
    2. **Selecting and filtering elements**
       1. Filter
          1. Tests each element and return true or false
    3. **Joining Data**
    4. **Node ordering**
       1. Raise()
       2. Lower()
    5. **Calling Functions from a selection**
       1. Use .each and .call to interrupt order of selection rules

## D3 general update pattern

## Loading and Parsing Data

* + 1. **Using delimiter separated values**
       1. Parserows
       2. Formatrows
    2. **Load and parse at same time**
       1. All functions are js promises

## Complete step by step example – join()

# Manipulating Data and Formatting

## Manipulating arrays collections and objects

* + 1. **Searching and Statistics**
       1. Min, max,extent,sum,mean,median,quantile,variance,deviation
    2. **Sorting**
       1. Descending – large to small
       2. Ascending – small to large
    3. **Transform Data**
    4. **Generating Data**
       1. Range, ticks, tickstep, tickincrement
    5. **Grouping Operations**
       1. Group
          1. Group objects by selected keys
          2. Can group by multiple keys
       2. Rollup
          1. Reduce each group by computing a summary value such as count or sum
          2. Data, reducer, key1, key2 …

## Random Numbers

## Interpolation

## Number Formatting

## Date and time parsing and formatting

# Scales Axes and Colors

## Axes

* + 1. **axisTop, bottom, left, right (scale)**
    2. **Ticks**
       1. tickValues, tickPadding, tckSizeInner, tickSizeOuter
       2. a g object that contains a line and a text element
       3. .tick line clss
       4. .domain is the line not the ticks
    3. **Styling**
    4. **Cartesian Axes**
       1. Yaxis
          1. (marginX, 0)
       2. XAxis
          1. (0, height – marginY)
       3. ScaleX
          1. [marginX, width – marginX]
       4. ScaleY
          1. [height – marginY, marginY]
       5. Xaxis label
          1. [(width/2),(marginY - labelPaddingY)]
       6. YAxis Label
          1. [-(marginX - labelPaddingX),(height/2)] and rotate 90
    5. **Radial Axes**

## Scales

* + 1. **Continuous Scales**
       1. Linear, pow (exponent), sqrt, log, symlog, identity, time, utc
       2. Domain- all possible input data values
       3. Range – should be mapped to how big pixel wise
       4. Invert – receives value from range and gives the domain value
       5. Clamp – if scale receives value outside domain then will return upper or lower limit
       6. Unknown- set output for unknown values
       7. *Linear Scale*
          1. Can be done with multiple segments value between each pair of domain values are mapped to corresponding pair in the range
       8. *Exponential Scale*
          1. *d3.scalePow()*
          2. default exponent is 1
          3. exponent should be provided with *exponent()* method
          4. Exponents with value less than 1 are rrot functions

Square root => *exponent(0.5)*

OR use *d3.scaleSqrt()*

* + - 1. *Logarithmic Scale*
         1. D3.scaleLog()
         2. Domain of log does NOT include zero
         3. Default base is 10
         4. Other bases => base() method
      2. *Symlog Scale*
         1. Symmetrical log scale d3.scaleSymlog()
         2. Mixture of 3 functions
         3. In the middle is a linear function that includes zero inside a small interval and a pair of symmetric logs from interval to infinity
      3. *Identity Scale*
         1. *D3.scaleIdentity()*
         2. Domain and range are identical
      4. *Time Scale*
         1. *D3.scaleTime()*
         2. *D3.scaleUtc()*
         3. Linear scale with temporal domain
         4. Domain take DATE objects
    1. **Interpolated Scales**
       1. Map a continuous domain to a fixed range that’s used by an interpolator to produce the output
       2. Two types: Sequential and Diverging
          1. Sequential

Unidirectional domains

* + - * 1. Diverging

Symmetrical domains

Extreme high, extreme low, and neutral middle

* + 1. **Discrete Scales**
       1. Map a domain which can be discrete or continuous input to a discrete range
       2. *Quantize Scale*
          1. Maps value of a continuous domain to a discrete range by dividing the domain into equal parts according to the # of items in the range
          2. Ex

Value domain [0, 900]

Mapped to one of three colors

<300 => c1 >300 < 600 => c2 > 600 => c3

* + - 1. *Quantile Scale*
         1. Maps a sampled set of discrete input values to a discrete range by dividing the sample into equal sized sets
         2. EX

12 item sample mapped to a 3 item range

* + - 1. *Threshold Scale*
         1. Domain is divided into subsets based on a list of n threshold values which are mapped toa discrete range of size n+1
      2. *Ordinal Scale*
         1. One to one mapping between discrete values
         2. If domain is larger than range than range will be repeated
      3. *Band Scale*
         1. Divides an output range into discrete bands of equal width and padding based on a list of discrete input categories
      4. *Point Scale* 
         1. Fits a number of points that are equally spaced within an interval

## Color Palettes, schemes, and spaces

* + 1. Color Palettes
       1. Palette – fixed size sequence of colors and is usually rep as an array in JS
       2. Scheme – a collection of color palettes and is usually rep as an object or function
       3. Use a scheme to generate a palette containing an arbitrary seq of colors
       4. ColorBrewer \*\*\*
    2. Categorical Color Schemes – d3.scheme\*
    3. Color Spaces
       1. D3.color