Interactive Data Visualization for the Web

An Introduction to Designing with D3

Goal

To give you a brief introduction to one way of visualization data on the web using d3.js

- What is d3 and why use it?
- How to get and process data for use
- How to create a chart to display the data
- How to add interactivity
- How to make it real time

Why Data Visualization?

Because it persuades, and makes it fast to process information for humans.

Also, people will pay for it.

How to visualize data? Mapping

The key idea is mapping data properties to some relational visual metric

Eg. number of things = height of bar, radius or radians of circle, time on x-axis

What is D3, and why use it?

“**D**ata **D**riven **D**ocuments”, a play on WWW, W3

D3 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 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.

https://d3js.org/

Why make it yourself?

“A medium to support discovery must be capable of expressing novel thought. Just as we don’t use phrasal templates for composing the written word, we can’t be limited to chart templates for visualization or a drop-down of formulas for statistical analysis. We need more than configuration. We need the composition of primitives into creations of our own design.”

- Mike Bostock

d3 examples:

Cartography: http://bl.ocks.org/KoGor/5994804

https://github.com/d3/d3/wiki/Gallery

Layouts + Voronoi:
https://bl.ocks.org/dunderjeep/65366fb7a4946f3b8ef9dbef2a63951b

Work-in-progress: https://graphs-e23a2.firebaseapp.com/

A little bit about the creator of d3 Mike Bostock

(Argument by authority: He's pretty smart) Read his blog: https://bost.ocks.org/mike/

Michael "Mike" Bostock is an American computer scientist and data-visualisation specialist. He is noted as one of the key developers of D3.js,[1] a JavaScript library used for producing dynamic, interactive, online data visualizations.[2] He was also involved in the preceding Protovis framework.

Bostock was a PhD student at Stanford University, advised by Jeffrey Heer.[3][4] Until 2015 he was working for the New York Times where he was leading complex data-visualisation projects.[5][6] Also for the New York Times he helped to create interactive articles.[7] He is also an adviser to data transformation platform provider Trifacta.

Bostock was interviewed in the Data Stories podcast and presented at Eyeo 2014.[1] The "Innovation Report" of his employer, the New York Times, called him a "digital superstar"[8] and Edward Tufte predicted that he will become one of the most important people for the future of data visualisation.[9]

A little bit about me

- went to school for computers; learned that everything is a tree
- work in e-commerce, sales and product information
- d3 is a hobby that I hope to bring into work - BI : Business Intelligence

The purpose of visualization is insight, not pictures.

Overview of demo

Let’s make an interactive scatterplot

- get and process the data
- display the data
  - create graph
  - add axis
  - add data
- add interactivity

How to get the data?

d3.csv(“path-to-data”, *optional-accessor-function,* function(data) {

```
    // do something with the data, like chart it
});
```

Other d3 data functions:

d3.tsv(), d3.json(), d3.request(), d3.defer()

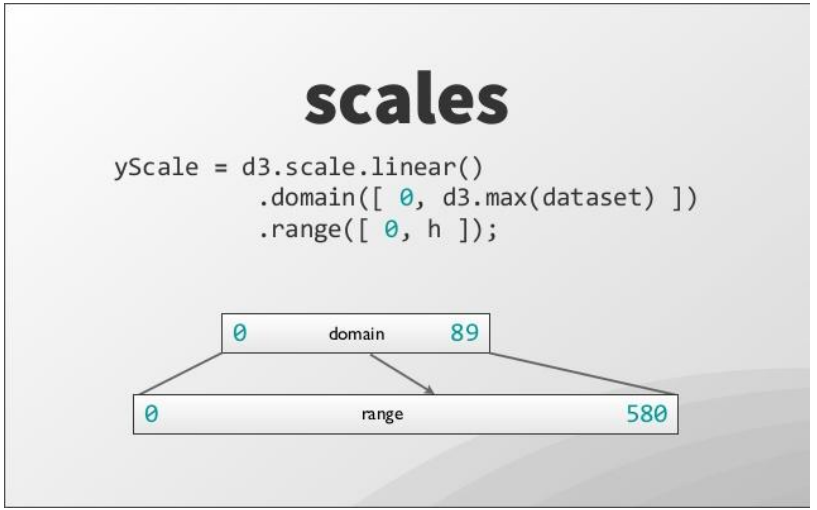
How to map the data to your graph? Scales

Scales are functions that map elements from an input domain to an output range

Input domain : the range of possible input data values

Output range : the range of possible output values (typically pixels)

(“Normalization” just means range is 0 to 1)



D3 has lots of ways to scale

- scaleLinear()
- timeScale()
- nice()
- rangeRound()
- scaleLog
- scaleQuantile

...

Scale helpers: d3.min(),d3.max(), d3.extent()

d3.max(data, *accessor function*);

```
d3.max(dataset, function() {
        return d[0];
});
```

Axes

d3.<type-of-axis>(<scale>);

eg. d3.axisBottom(x);

How to bind the data to graph? Selections

```
d3.select(“body”).selectAll(“p”)  // returns d3, reference to body, reference to p
    .data([1,2,3,4,5])  // reference to dataset, counts data
    .enter()  // magic!  looks at last selection, and the data
    .append(“p”)
    .text(function(d) { return “My number is: ” + d});
```

The most common purpose of creating a selection is ultimately to modify it in some way, such as using attr() or style().

How to add interactivity? Event Listeners, transitions

d3.select(“p”).on(“click”, function() { alert(“click”); });

Interactivity is most powerful when it can provide different views of data, letting the user explore the information from different angles.

Examples of different views:

- *sorting*
- *magnify*
- *relationships*

Transitions

.duration(1000) // how long it takes

.ease(d3.easelinear)

.on(‘start’)

Tooltips

Using HTML tooltip, you can style it however you want, even link it to url

How to make it real time? Updates

Enter, Exit, Merge, and Update

More: Layouts, database integration

https://graphs-e23a2.firebaseapp.com/

https://console.firebase.google.com/u/0/project/graphs-e23a2/database/graphs-e23a2/data

Can I export my graph?

Yes

Sharing Your Code? Use blocks

Blocks : https://bl.ocks.org/

My block: https://bl.ocks.org/dunderjeep/65366fb7a4946f3b8ef9dbef2a63951b

Responsible Chart Use

https://guides.library.duke.edu/datavis/topten

https://www.ibm.com/design/language/experience/data-visualization/chart-modelsif

Data visualization Tips

- Show relationships: Performance vs goals comparison -
- Line Charts track changes or trends over time and show the relationship between two or more variables.
- Bar Charts are used to compare quantities of different categories.
- Scatter Plots show joint variation of two data items.
- Bubble Chart show joint variation of three data items.
- Pie Charts are used to compare parts of a whole and should be used carefully.
- Never compare two pie charts without clearly noting that the size of the pie may have changed as well.

8 Golden Rules of Interface Design

1 Strive for consistency.
Consistent sequences of actions should be required in similar situations; identical terminology should be used in prompts, menus, and help screens; and consistent commands should be employed throughout.

2 Enable frequent users to use shortcuts.
As the frequency of use increases, so do the user's desires to reduce the number of interactions and to increase the pace of interaction. Abbreviations, function keys, hidden commands, and macro facilities are very helpful to an expert user.

3 Offer informative feedback.
For every operator action, there should be some system feedback. For frequent and minor actions, the response can be modest, while for infrequent and major actions, the response should be more substantial.

4 Design dialog to yield closure.
Sequences of actions should be organized into groups with a beginning, middle, and end. The informative feedback at the completion of a group of actions gives the operators the satisfaction of accomplishment, a sense of relief, the signal to drop contingency plans and options from their minds, and an indication that the way is clear to prepare for the next group of actions.

5 Offer simple error handling.
As much as possible, design the system so the user cannot make a serious error. If an error is made, the system should be able to detect the error and offer simple, comprehensible mechanisms for handling the error.

6 Permit easy reversal of actions.
This feature relieves anxiety, since the user knows that errors can be undone; it thus encourages exploration of unfamiliar options. The units of reversibility may be a single action, a data entry, or a complete group of actions.

7 Support internal locus of control.
Experienced operators strongly desire the sense that they are in charge of the system and that the system responds to their actions. Design the system to make users the initiators of actions rather than the responders.

8 Reduce short-term memory load.
The limitation of human information processing in short-term memory requires that displays be kept simple, multiple page displays be consolidated, window-motion frequency be reduced, and sufficient training time be allotted for codes, mnemonics, and sequences of actions.