CHAPTER 2

Introducing D3

D3—also referred to as d3.js—is a JavaScript library for creating data visualizations. But that kind of undersells it.

The abbreviation D3 references the tool's full name, *Data-Driven Documents*. The *data* is provided by you, and the *documents* are web-based documents, meaning anything that can be rendered by a web browser, such as HTML and SVG. D3 does the *driving*, in the sense that it connects the data to the documents.

Of course, the name also functions as a clever allusion to the network of technologies underlying the tool itself: the W3, or World Wide Web, or, today, simply "the web."

D3's primary author is the brilliant Mike Bostock, although there are several other dedicated contributors. The project is entirely open source and freely available on GitHub.

D3 is released under a BSD license, so you may use, modify, and adapt the code for noncommercial or commercial use at no cost.

D3's official home on the Web is *d3js.org*.

What It Does

Fundamentally, D3 is an elegant piece of software that facilitates generation and manipulation of web documents with data. It does this by:

- *Loading* data into the browser's memory
- *Binding* data to elements within the document, creating new elements as needed

- *Transforming* those elements by interpreting each element's bound datum and setting its visual properties accordingly
- *Transitioning* elements between states in response to user input

Learning to use D3 is simply a process of learning its syntax, so you can tell it how to load and bind data, and transform and transition elements.

The *transformation* step is most important, as this is where the *mapping* happens. D3 provides a structure for applying these transformations, but, as we'll see, you define the mapping rules. Should larger values make taller bars or brighter circles? Will clusters be sorted on the x-axis by age or category? What color palette is used to fill in countries on your world map? All of the visual design decisions are up to you. You provide the concept, you craft the rules, and D3 executes it—without telling you what to do.

What It Doesn't Do

Here is a list of things D3 does not do:

- D3 doesn't generate predefined or "canned" visualizations for you. This is on purpose. D3 is intended primarily for highly customized visualization work, whether that is designing one-off explanatory charts or complex, interactive, exploratory tools. It is the most powerful tool for visualization on the web specifically because it enables you to develop whatever you can imagine from scratch. There are no templates or chart "wizards" in D3 (although you may become one by the time you finish this book). There are, however, many excellent tools built on top of D3 that *do* provide access to preconfigured chart types. (See the section "Alternatives" on page 10.)
- D3 doesn't even try to support older browsers. This helps keep the D3 codebase clean and free of hacks to support old versions of Internet Explorer, for example. The philosophy is that by creating more compelling tools and refusing to support older browsers, we encourage more people to upgrade (rather than forestall the process, thereby requiring us to continue to support those browsers, and so on—a vicious cycle). When D3 was first released in 2011, this was a fairly radical position. I'm happy to say that browsers, people, and organizations have since modernized sufficiently to the point that this is practically a nonissue. Bureaucracies that continue to support ancient browsers miss out on all the benefits of D3. (More for us!)
- D3's *core* functionality doesn't handle bitmap map tiles, such as those provided by Google Maps or Mapbox. D3 is great with anything vector—SVG images or GeoJSON data—but wasn't originally intended to work with traditional map tiles. (*Bitmap* images are made up of pixels, so resizing them larger or smaller is diffi-

cult without a loss in quality. *Vector* images are defined by points, lines, and curves—mathematical equations, really—and can be scaled up or down without a loss in quality.) Fortunately, the d3-tile plug-in can be used for tile-based mapping, though it is not covered in this book.

- D3 doesn't hide your original data. Because D3 code is executed on the client side (meaning, in the user's web browser, as opposed to on the web server), the data you want visualized must be sent to the client. If your data can't be shared, then don't use D3. Alternatives include using proprietary browser plug-ins (like Flash) or prerendering visualizations as static images and sending those to the browser. (Yet, if you're not interested in sharing your data, why would you bother visualizing it? The purpose of visualization is to communicate the data, so you might sleep better at night by choosing openness and transparency, rather than having nightmares about data thieves.)

Origins and Context

The first web browsers rendered static pages; interactivity was limited to clicking links. In 1996, Netscape introduced the first browser with JavaScript, a new scripting language that could be interpreted *by the browser while the page was being viewed*.

This doesn't sound as groundbreaking as it turned out to be, but this enabled web browsers to evolve from merely passive *browsers* to dynamic frames for interactive, networked experiences. This shift ultimately enabled every intrapage interaction we have on the web today. Without JavaScript, D3 would never have existed, and web-based data visualizations would be limited to prerendered, noninteractive GIFs. (Yuck. Thank you, Netscape!)

Jump ahead to 2005, when Jeffrey Heer, Stuart Card, and James Landay introduced prefuse, a toolkit for bringing data visualization to the web. prefuse (spelled with all lowercase letters) was written in Java, a compiled language, with programs that could run in web browsers via a Java plug-in. (Note that *Java* is a completely different programming language than *JavaScript*, despite their similar names.)

prefuse was a breakthrough application—the first to make web-based visualization accessible to less-than-expert programmers. Until prefuse came along, any data vis on the web was very much a custom affair.

Two years later, Jeff Heer introduced Flare, a similar toolkit, but written in ActionScript, so its visualizations could be viewed on the web through Adobe's Flash Player. Flare, like prefuse, relied on a browser plug-in. Flare was a huge improvement, but as web browsers continued to evolve, it was clear that visualizations could be created with native browser technology, no plug-ins required.

By 2009, Jeff Heer had moved to Stanford, where he was advising a graduate student named Michael Bostock. Together, in Stanford's Vis Group, they created Protovis, a JavaScript-based visualization toolkit that relied exclusively on native browser technologies.

Protovis made generating visualizations simple, even for users without prior programming experience. Yet, to achieve this, it created an abstract representation layer. The designer could address this layer using Protovis syntax, but it wasn't accessible through standard methods, so debugging was difficult.

In 2011, Mike Bostock, Vadim Ogievetsky, and Jeff Heer officially announced D3, the next evolution in web visualization tools. Unlike Protovis, D3 operates directly on the web document itself. This means easier debugging, easier experimentation, and more visual possibilities. The only downside to this approach is a potentially steeper learning curve, but this book will make that as painless as possible. Plus, all the skills you gain while learning about D3 will prove useful even beyond the realm of data vis.

If you're familiar with any of these groundbreaking tools, you'll appreciate that D3 descends from a prestigious lineage. And if you have any interest in the philosophy underlying D3's elegant technical design, I highly recommend Mike, Vadim, and Jeff's InfoVis paper, which clearly articulates the need for this kind of tool. The paper encapsulates years' worth of learning and insights made while developing visualization tools.

Alternatives

D3 might not be perfect for every project. Sometimes you just need a quick chart and you don't have time to code it from scratch. Or you might need to support *really* old browsers and can't rely on the presence of technologies like SVG.

For those situations, it's good to know what other tools are out there. Here is a brief, noncomprehensive list of D3 alternatives, all of which use web-standard technologies (mostly JavaScript) and are free to download and use.

Easy Charts

DataWrapper
: A beautiful web service that lets you upload your data and quickly generate a chart that you can republish elsewhere, embed on your site, or export to PDF. This service was originally intended for journalists, but it is helpful for everyone. DataWrapper displays interactive charts in current browsers and static images for old ones. (Brilliant!) You can also download all the code and run it on your own server instead of using theirs.

Flot
: A plotting library for jQuery that uses the HTML canvas element and supports older browsers, even all the way back to Internet Explorer 6. It supports limited visual forms (lines, points, bars, areas), but it is easy to use.

Google Chart Tools
: Having evolved from Google's earlier Image Charts API, Google's Chart Tools can be used to generate several standard chart types, with support for old versions of IE.

Highcharts
: A JavaScript-based charting library with several predesigned themes and chart types. The tool is free only for noncommercial use.

Peity
: A jQuery plug-in for very simple and very *tiny* bar, line, and pie charts that supports only recent browsers. Did I mention that this makes only very *tiny* visualizations? +10 cuteness points.

Timeline.js
: A library specifically for generating interactive timelines. No coding is required; just use the code generator. There is not much room for customization, but hey, timelines are really hard to do well.

Graph Visualizations

A "graph" is just data with a networked structure (for example, B is connected to A, and A is connected to C).

Arbor.js
: A library for graph visualization using jQuery. Even if you never use this, you should check out how the documentation is presented as a graph, using the tool itself. (It's so *meta*.) It uses the HTML canvas, so it won't work in older browsers.

Cytoscape.js
: Library for graph theory analysis and visualization.

Sigma.js
: A very lightweight library for graph visualization. Sigma.js is beautiful and fast, and it also uses canvas.

Geomapping

In this book, I distinguish between *mapping* (all visualizations are maps) and *geomapping* (visualizations that include geographic data, or geodata, such as traditional

maps). D3 has a lot of geomapping functionality, but you should know about these other tools.

Kartograph
: A JavaScript-and-Python combo for gorgeous, entirely vector-based mapping by Gregor Aisch with must-see demos. Please go look at them now. I promise you've never seen online maps this beautiful.

Leaflet
: A library for tiled maps, designed for smooth interaction on both desktop and mobile devices. It includes some support for displaying data layers of SVG on top of the map tiles. (See Mike's demo "Using D3 with Leaflet".)

Modest Maps
: The granddaddy of tiled map libraries, Modest Maps has been succeeded by Polymaps and D3, but lots of people still love it, as it is lightweight and works with old versions of IE and other browsers. Modest Maps has been adapted for ActionScript, Processing, Python, PHP, Cinder, openFrameworks…yeah, basically everything. File this under "oldie, but goodie."

Polymaps
: A predecssor of D3 by Mike Bostock, this library is for displaying tiled maps, with layers of data on top of the tiles. Polymaps relies on SVG and thus works best with current browsers. That said, you may be better off using D3 and the d3-tile plug-in.

Almost from Scratch

These tools, like D3, provide methods of drawing visual forms, but without predesigned visual templates. If you enjoy the creative freedom of starting from scratch, you might enjoy these.

p5.js
: p5 takes Processing, the fantastic programming language for artists and designers, and reimagines it in JavaScript for the web. Imagine the friendly nomenclature of Processing, and the webby strength of JavaScript. The p5 project is led by Lauren McCarthy, and it renders using canvas, so only modern browsers are supported.

Paper.js
: A framework for rendering vector graphics to canvas. Also, its website is one of the most beautiful on the internet, and their demos are unbelievable. (Go play with them now.)

Raphaël
: A well-established library for drawing vector graphics by Dmitry Baranovskiy, popular due to its friendly syntax and support for older browsers.

Snap.svg
: A pretty fantastic, modern library for SVG creating and animation, this is also primarily by Dmitry. Consider it Raphaël's successor.

Two.js
: JavaScript library for two-dimensional drawing in modern browsers, rendering to SVG, canvas, and WebGL, by Jono Brandel.

Three-Dimensional

D3 is not the best at 3D, simply because web browsers are historically two-dimensional beasts. But with increased support for WebGL, there are now more opportunities for 3D web experiences.

PhiloGL
: A WebGL framework specifically for 3D visualization (no longer under active development, unfortunately).

Three.js
: A library for generating any sort of 3D scene you could imagine, produced by Google's Data Arts team. You could spend all day exploring the mind-blowing demos on their site.

Tools Built with D3

General-use charting libraries

There are many different charting libraries built on top of D3. Theoretically, these make it easier to generate a visualization quickly, and often without having to write any D3 code yourself. The trade-off is generally less customization; you have to be comfortable with the chart templates supported by each tool. Also, each library has its own syntax and quirks. I recommend taking a quick glance at each one to decide what works best for you.

Britecharts
: A reusable charting library by for D3 4.x brought to you by Eventbrite's engineering team. And, wow, those colors sure are "brite."

C3.js
: Reusable charting by Masayuki Tanaka. Not yet updated to work with D3 4.x at the time of this writing. (But what a nice demo!)

Contour
: Beautifully designed, simple chart types.

D3plus
: Charting library that also includes some nice utilities for easy text wrapping, color legibility, and other things you'd probably want help with.

D4
: Library with lots of supported chart types.

dimple
: Library intended for business analysts.

NVD3
: NVD3 was one of the first D3-based charting libraries, and offers lots of beautiful examples, with room for customization.

plotly.js
: Quick and easy charting. Just drop in your data values, and you're off!

Plottable
: Promises "the power and flexibility of D3, but easier," by providing predefined "components" that you can reuse.

uvCharts
: Another such library, with 12 supported chart types.

Vega
: A "visualization grammar" with which you define chart types, visual properties, interaction rules, and data in a simple JSON object (more on JSON in Chapter 3, *Technology Fundamentals*). Then Vega translates your specifications into a working, interactive chart, using D3 under the hood. Version 2 of this amazing, powerful tool was primarily authored by Arvind Satyanarayan, and was produced in Jeff Heer's new Interactive Data Lab at the University of Washington (Jeff's next stop after Stanford).

More specialized tools

This section includes D3-based libraries with more specialized use cases (such as for time series data), as well as plug-ins for use with D3 and other related tools.

Crossfilter
: A library for working with large, multivariate datasets, written primarily by Mike Bostock. This is useful for trying to squeeze your "big data" into a relatively small web browser. Not technically built with D3, but is commonly used with D3.

Cubism
: A D3 plug-in for visualizing time series data, also written by Mike Bostock. (One of my favorite demos.)

d3-annotation
: A module for painlessly implementing visual annotations in D3 by Susie Lu.

d3-context-menu
: A plug-in for adding contextual menus to your D3 projects, by Patrick Gillespie.

d3.sketchy
: This tool by Sebastian Meier takes your SVG shapes and makes them look hand-drawn. Useful when you are working in code, but need to convey to others that the output is rough and your design is still in process (like a sketch). Be sure to play with the interactive customizer.

D3 SVG Legend
: A reusable legend component for D3 by Susie Lu.

D3-tip
: A tool for generating tooltips in D3 charts, in case you get tired of making your own, as described in Chapter 10.

D6
: To be honest, I don't understand this one, but I had to share it here because the name stands for "Dynamically Downloaded Data-Driven Documents, Dude."

dc.js
: The "dc" is short for *dimensional charting*, as this library is optimized for exploring large, multidimensional datasets.

Firespray
: Super-fast charting library for streaming data. (Think high-density real-time data dashboards.)

Forest D3
: A time-series charting library built on D3, by Robin Hu.

MetricsGraphics.js
: A very nice library for working with time-series data, by Ali Almossawi and Hamilton Ulmer.

Miso Project
: An open source project that includes d3.chart, "a framework for building reusable charts with d3.js," as well as other useful tools, from the brilliant people at Bocoup and The Guardian Interactive team.

RAW Graphs
: Paste your spreadsheet into this amazing tool and generate an array of different chart types in seconds. A project initiated at the esteemed Density Design research lab in Milan.

R2D3
: A unique blend of D3 and R that enables you to use R to create D3 visualizations.

Rickshaw
: A toolkit for displaying time series data that is also very customizable.

TechanJS
: A library specifically for financial data charting and analysis.

Tributary
: A great tool for experimenting with live coding using D3, by Ian Johnson.