Interactive Data Visualisation

Data Visualisation & D3

Dr Ruairi O'Reilly

More literally, visualization is a process of mapping information to visuals. We craft
rules that interpret data and express its values as visual properties.

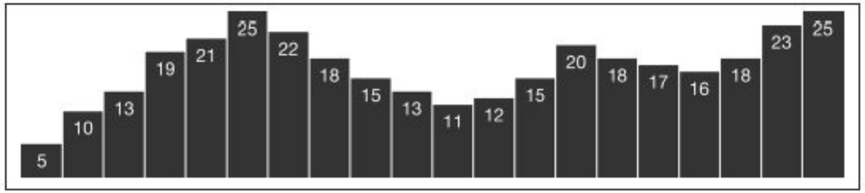
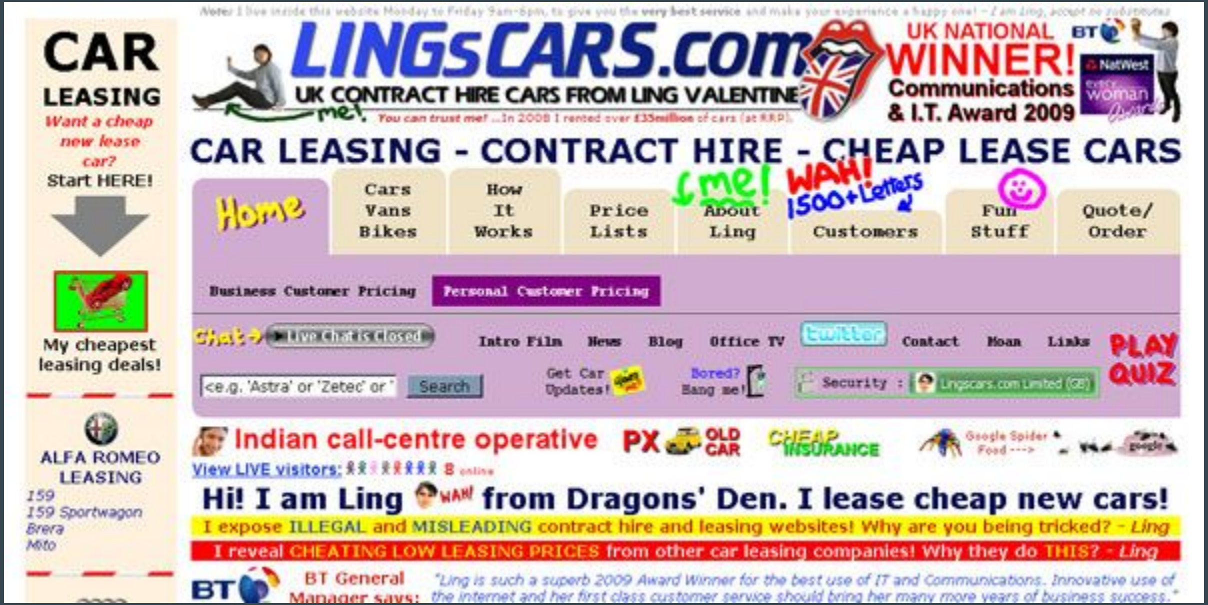


Figure 1-1. Data values mapped to visuals

Data Visualisation

- Information overload. Excess amounts of information are overwhelming; raw data becomes useful only when we apply methods of deriving insight from it.
- Humans are intensely visual creatures. Few of us can detect patterns among rows of numbers, but even young children can interpret bar charts.
- Visualizing data is the fastest way to communicate it to others.
- Visualizations, like words, can be used to lie, mislead, or distort the truth.

Information Overload



Note: I live inside this website Monday to Friday 9am-6pm, to give you the **very best service** and make your experience a happy one! – *I am Ling, accept no substitutes*

LINGsCARS.com

UK CONTRACT HIRE CARS FROM LING VALENTINE

me!

You can trust me! ...In 2008 I rented over £35million of cars (at RRP).

UK NATIONAL WINNER! Communications & I.T. Award 2009

CAR LEASING - CONTRACT HIRE - CHEAP LEASE CARS

CAR LEASING

Want a cheap new lease car?

Start HERE!

My cheapest leasing deals!

ALFA ROMEO LEASING

159
159 Sportwagon
Brera
Mito

Indian call-centre operative

View LIVE visitors: 8 online

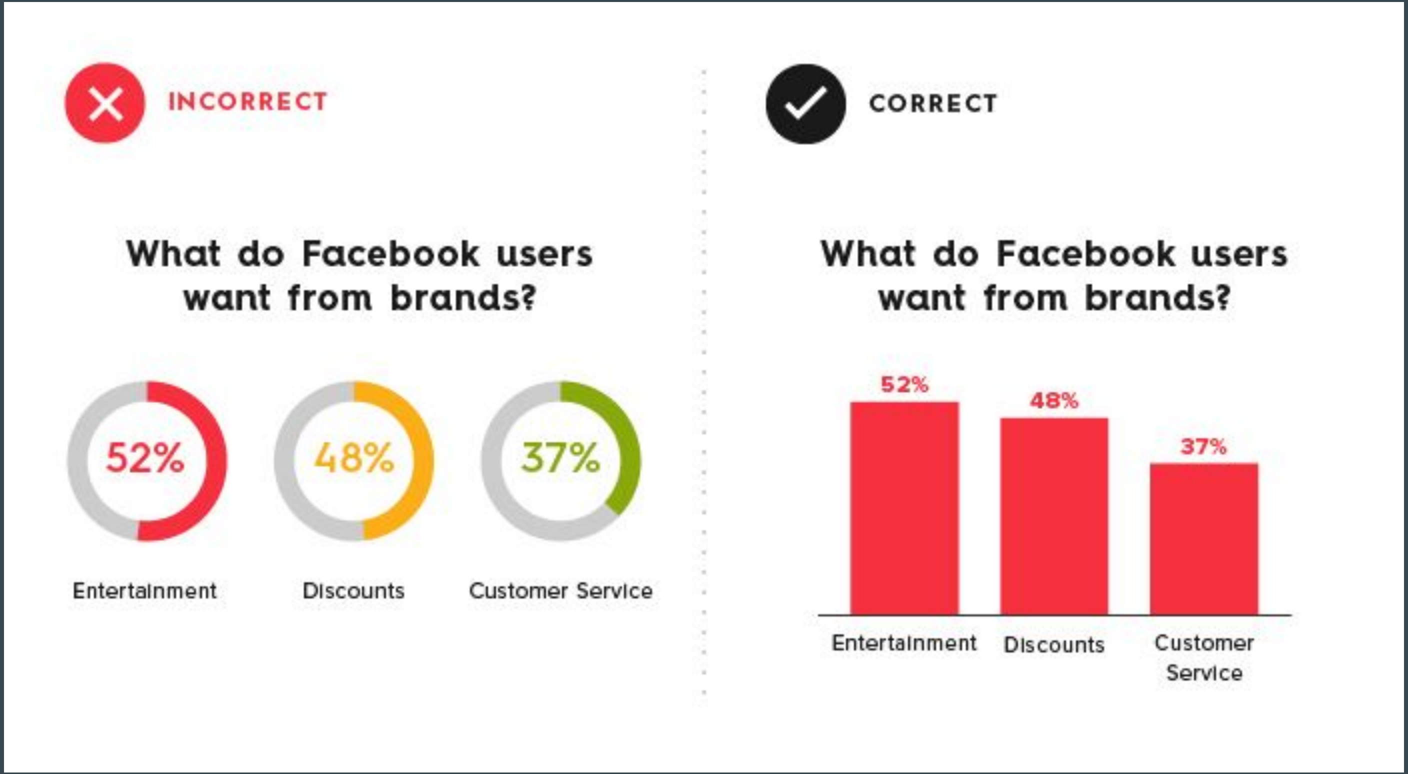
Hi! I am Ling WAH! from Dragons' Den. I lease cheap new cars!

I expose ILLEGAL and MISLEADING contract hire and leasing websites! Why are you being tricked? - *Ling*

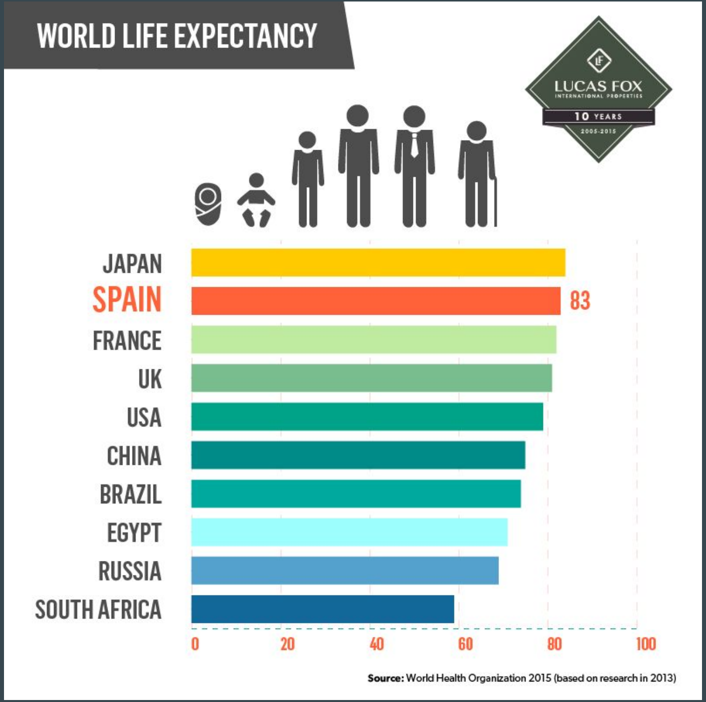
I reveal CHEATING LOW LEASING PRICES from other car leasing companies! Why they do THIS? - *Ling*

BT General Manager says: *"Ling is such a superb 2009 Award Winner for the best use of IT and Communications. Innovative use of the internet and her first class customer service should bring her many more years of business success."*

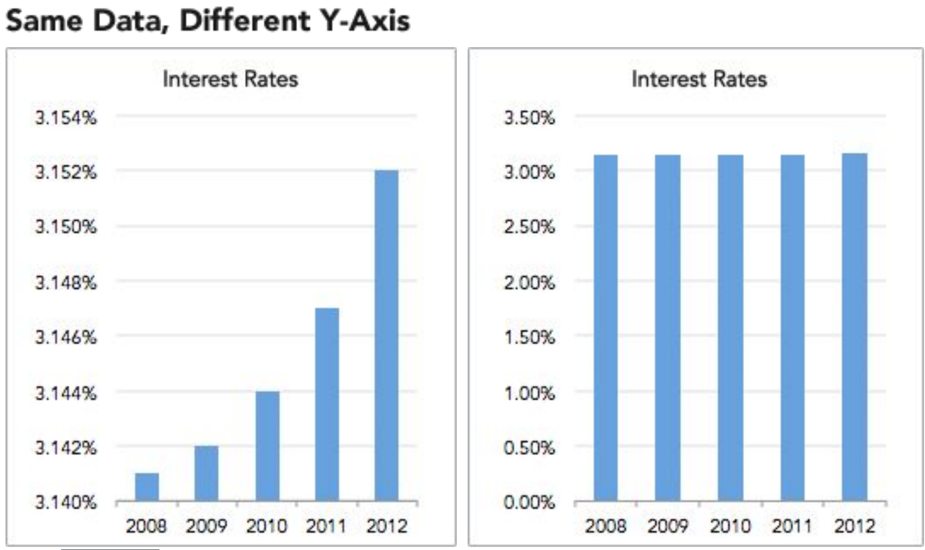
Humans are intensely visual creatures



Visualizing data and communication



Visualizations, like words, can be used to lie, mislead, or distort the truth



Why Write Code?

- Automation
- Greater quantities of data (reduced man hours)
- Experimentation

- Sets of mapping rules function as design systems. The human hand no longer executes the visual output; the computer does. Our human role is to conceptualize, craft, and write out the rules of the system, which is then finally executed by software.

Why interactive?

- Variety of perspectives
- Dimensionality of information
- Representing multidimensional datasets fairly in static images is notoriously difficult.
- A fixed image is ideal when alternate views are neither needed nor desired, and required when publishing to a static medium, such as print.

Dynamic, interactive visualizations can empower people to explore the data for themselves.

1996 - Ben Shneiderman of the University of Maryland first proposed a “Visual Information-Seeking Mantra”: overview first, zoom and filter, then details on demand.

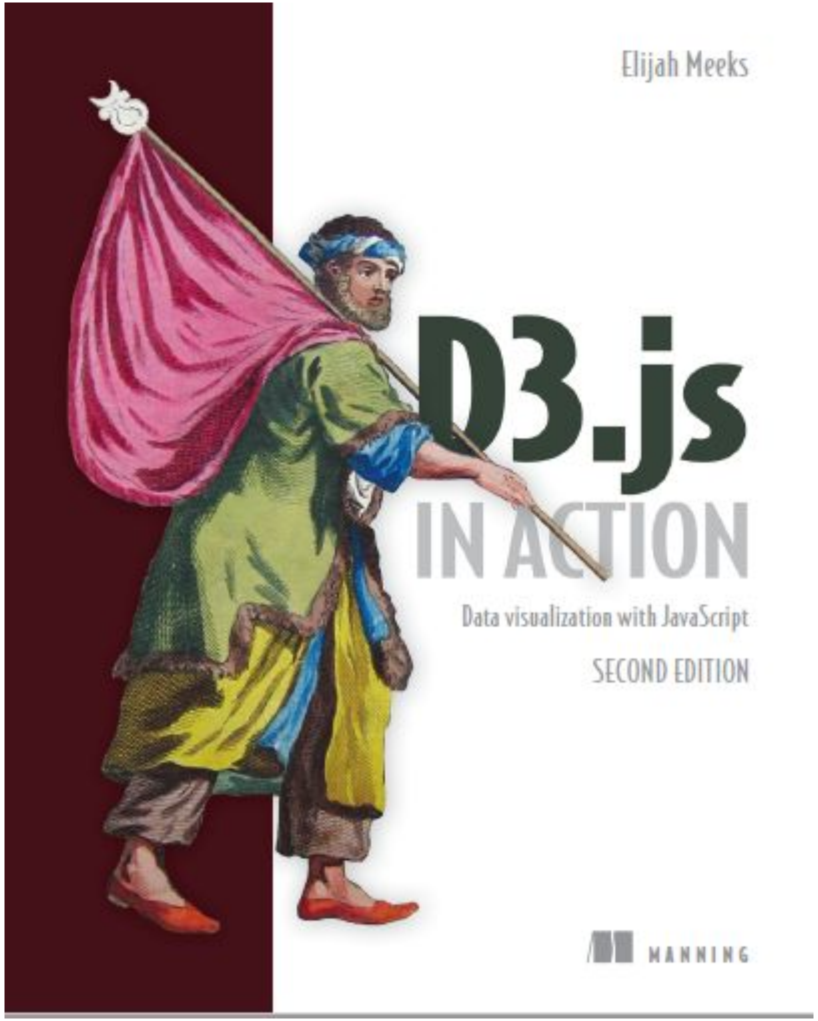
This design pattern is found in most interactive visualizations today. The combination of functions is successful, because it makes the data accessible to different audiences, from those who are merely browsing or exploring the dataset to those who approach the visualization with a specific question in search of an answer. An interactive visualization that offers an overview of the data alongside tools for “drilling down” into the details may successfully fulfill many roles at once

Why on the Web?

Visualizations aren’t truly visual unless they are seen.

D3.js in Action

2nd Edition, Elijah Meeks



Elijah Meeks

D3.js IN ACTION

Data visualization with JavaScript

SECOND EDITION

MANNING

Introducing D3

D3 is a JavaScript library for creating data visualizations.

D3 - 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.

The project is entirely open source and freely available on GitHub and released under a BSD license.

https://d3js.org/ List of online tutorials

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

Simply a process of learning the syntax used to tell it how you want it 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. (Yes, it's like the opposite of Excel's pushy "Chart Wizard.")

What It Doesn’t Do

D3 doesn’t generate predefined or “canned” visualizations for you.

D3 is intended primarily for explanatory visualization work, as opposed to exploratory visualizations.

Exploratory tools help you discover significant, meaningful patterns in data. e.g. Tableau, ggplot2

That’s an essential step, but different from generating an explanatory presentation of the data, a view of the data that highlights what you’ve already discovered.

Explanatory views are more constrained and limited, but also focused, and designed to communicate only the important points. D3 excels at this latter step, but is not ideal for the former.

What It Doesn’t Do

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). D3 wants us to move forward.

D3’s core functionality doesn’t handle bitmap map tiles

E.g. Google Maps or Cloudmade.

- D3 is great with anything vector!
- This is starting to change, with the introduction of the d3.geo.tile plug-in. Prior to this plug-in, geomapping with D3 meant either going all-SVG and avoiding tiles or using D3 to create SVG visuals on top of a base layer of map tiles.

This question of how to integrate bitmap tiles and vector graphics comes up a lot in the D3 community. No super-simple and perfect answer, change on the way.

What It Doesn’t Do

D3 doesn’t hide your original data. Because D3 code is executed on the client side, the data you want visualized must be sent to the client.

If your data can’t be shared, then don’t use D3.

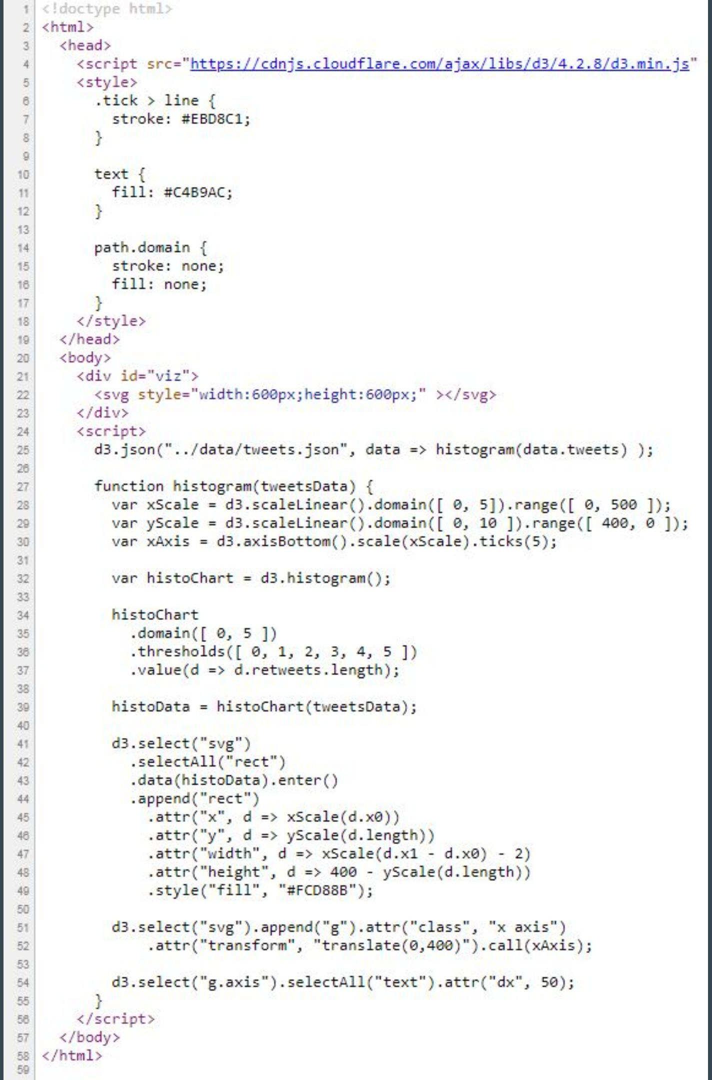
Alternatives include using proprietary tools (like Flash) or prerendering visualizations as static images and sending those to the browser.

Some examples

(running on my local machine)

A Basic Example - Bar Chart

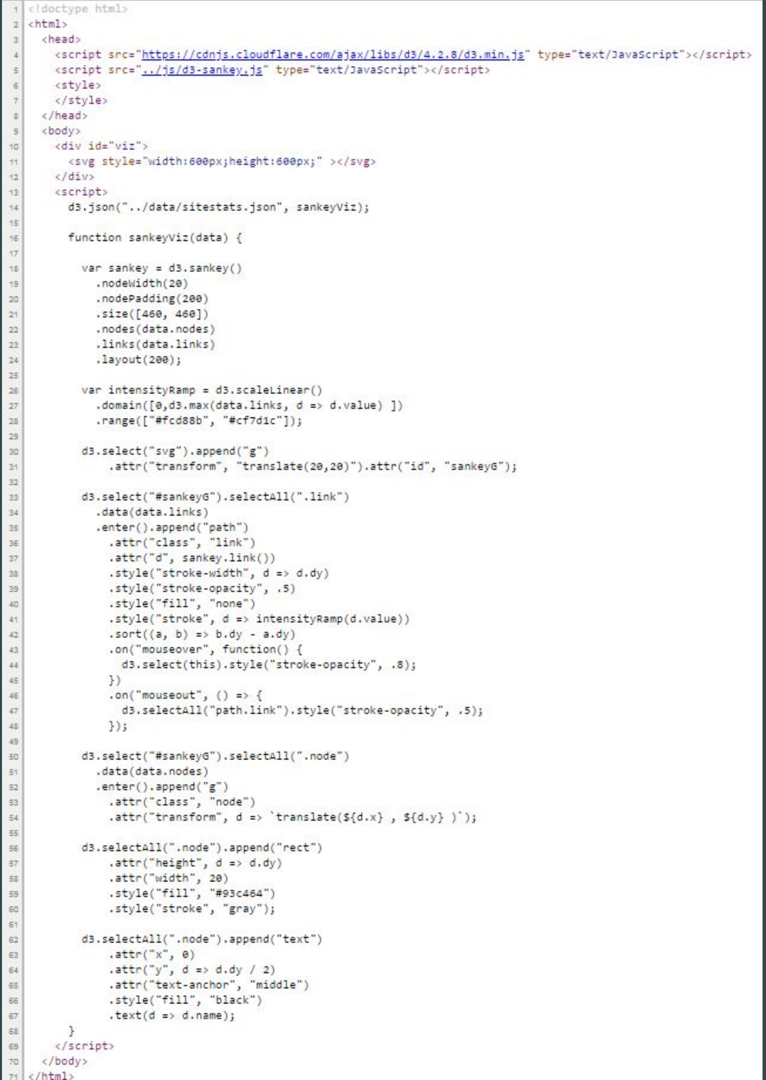
Bar Chart



```
<!doctype html>
<html>
  <head>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/d3/4.2.8/d3.min.js"
    <style>
      .tick > line {
        stroke: #EBD8C1;
      }

      text {
        fill: #C4B9AC;
      }

      path.domain {
        stroke: none;
        fill: none;
      }
    </style>
  </head>
  <body>
    <div id="viz">
      <svg style="width:600px;height:600px;" ></svg>
    </div>
    <script>
      d3.json("../data/tweets.json", data => histogram(data.tweets) );

      function histogram(tweetsData) {
        var xScale = d3.scaleLinear().domain([ 0, 5]).range([ 0, 500 ]);
        var yScale = d3.scaleLinear().domain([ 0, 10 ]).range([ 400, 0 ]);
        var xAxis = d3.axisBottom().scale(xScale).ticks(5);

        var histoChart = d3.histogram();

        histoChart
          .domain([ 0, 5 ])
          .thresholds([ 0, 1, 2, 3, 4, 5 ])
          .value(d => d.retweets.length);

        histoData = histoChart(tweetsData);

        d3.select("svg")
          .selectAll("rect")
          .data(histoData).enter()
          .append("rect")
            .attr("x", d => xScale(d.x0))
            .attr("y", d => yScale(d.length))
            .attr("width", d => xScale(d.x1 - d.x0) - 2)
            .attr("height", d => 400 - yScale(d.length))
            .style("fill", "#FCD88B");

        d3.select("svg").append("g").attr("class", "x axis")
            .attr("transform", "translate(0,400)").call(xAxis);

        d3.select("g.axis").selectAll("text").attr("dx", 50);
      }
    </script>
  </body>
</html>
```

A Basic Example - Site Map

Site map



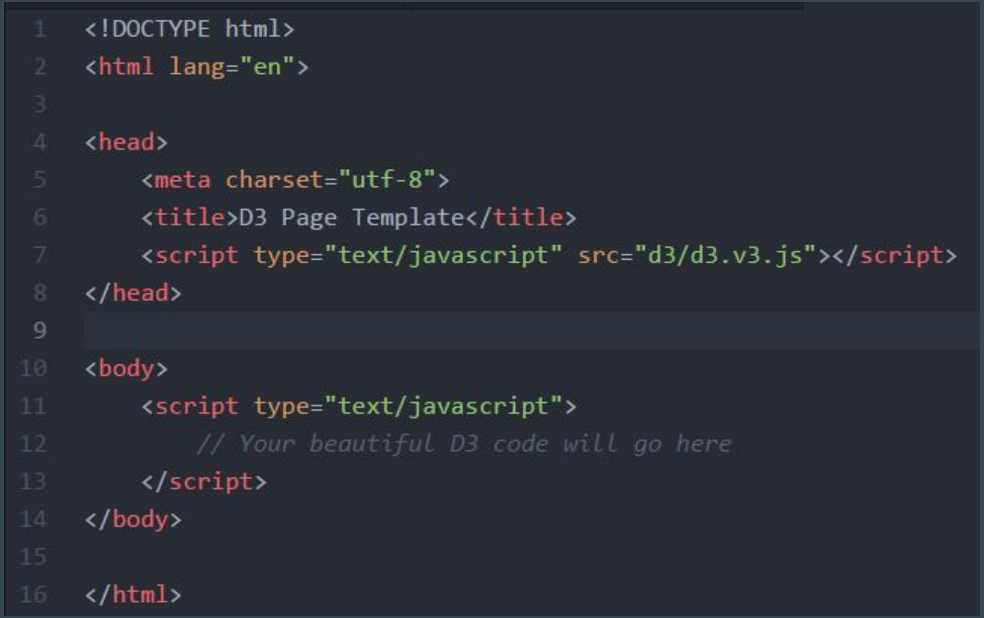
```
<!doctype html>
<html>
  <head>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/d3/4.2.8/d3.min.js" type="text/JavaScript"></script>
    <script src="../js/d3-sankey.js" type="text/JavaScript"></script>
    <style>
    </style>
  </head>
  <body>
    <div id="viz">
      <svg style="width:600px;height:600px;" ></svg>
    </div>
    <script>
      d3.json("../data/sitestats.json", sankeyViz);

      function sankeyViz(data) {

        var sankey = d3.sankey()
          .nodeWidth(20)
          .nodePadding(200)
          .size([460, 460])
          .nodes(data.nodes)
          .links(data.links)
          .layout(200);

        var intensityRamp = d3.scaleLinear()
          .domain([0,d3.max(data.links, d => d.value) ])
          .range(["#fcd88b", "#cf7d1c"]);

        d3.select("svg").append("g")
            .attr("transform", "translate(20,20)").attr("id", "sankeyG");

        d3.select("#sankeyG").selectAll(".link")
          .data(data.links)
          .enter().append("path")
            .attr("class", "link")
            .attr("d", sankey.link())
            .style("stroke-width", d => d.dy)
            .style("stroke-opacity", .5)
            .style("fill", "none")
            .style("stroke", d => intensityRamp(d.value))
            .sort((a, b) => b.dy - a.dy)
            .on("mouseover", function() {
              d3.select(this).style("stroke-opacity", .8);
            })
            .on("mouseout", () => {
              d3.selectAll("path.link").style("stroke-opacity", .5);
            });

        d3.select("#sankeyG").selectAll(".node")
          .data(data.nodes)
          .enter().append("g")
            .attr("class", "node")
            .attr("transform", d => `translate(${d.x} , ${d.y} )`);

        d3.selectAll(".node").append("rect")
            .attr("height", d => d.dy)
            .attr("width", 20)
            .style("fill", "#93c464")
            .style("stroke", "gray");

        d3.selectAll(".node").append("text")
            .attr("x", 0)
            .attr("y", d => d.dy / 2)
            .attr("text-anchor", "middle")
            .style("fill", "black")
            .text(d => d.name);
      }
    </script>
  </body>
</html>
```

More complex examples

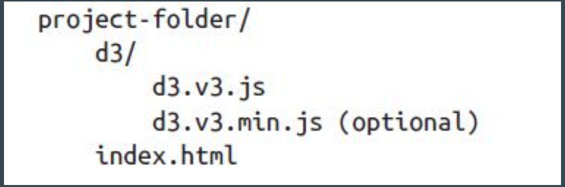
Starting point : The Globe	Finished product The Globe	(Chapter 8)
Starting point: dots	Finished product: Interactive dots	(Chapter 10)

Getting setup

- Download the latest version.
- Creating an empty page in which to write your code.
- Set up a local web server.



```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>D3 Page Template</title>
    <script type="text/javascript" src="d3/d3.v3.js"></script>
</head>

<body>
    <script type="text/javascript">
        // Your beautiful D3 code will go here
    </script>
</body>

</html>
```



```
project-folder/
    d3/
        d3.v3.js
        d3.v3.min.js (optional)
    index.html
```

A few things to note about this template:

- The meta tag identifies the encoding for this file as utf-8, which is needed to ensure that the browser can parse D3’s functions and data properly.
- The first script tag sets the reference to d3.js. You should edit this file path as needed if you’re using the minified version of D3 or decided to locate d3.js somewhere other than the d3 directory.
- The second script tag, in the body, is where you will soon key in all your beautiful code.

Editing a simple webpage with console

Using d3.select to set style and HTML content:

```
d3.select("body").append("div")
        .style("border", "1px black solid")
        .html("hello world");
```

Using d3.select to set attributes and event listeners

```
d3.select("div").style("background-color", "pink").style("font-size", "24px")
.attr("id", "newDiv").attr("class", "d3div").on("click", () => {console.log("You clicked a
div")});
```

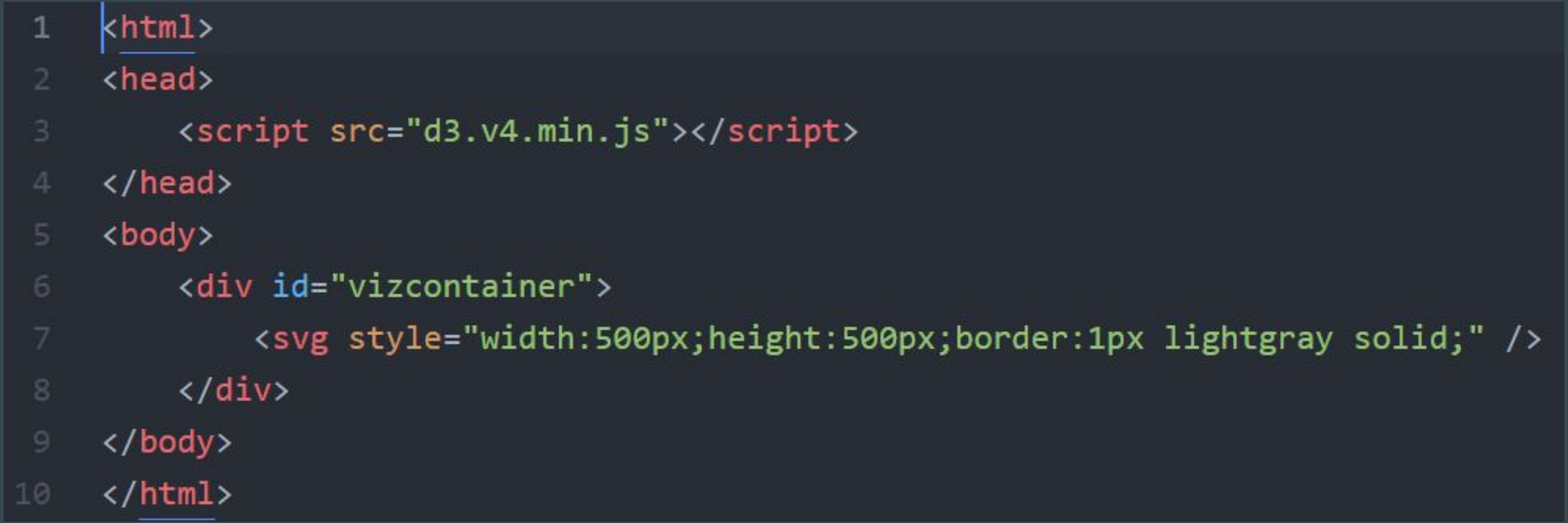
Hello World with circles

Your no here to learn how to add divs to a web page, but you likely want to deal with graphics like lines and circles. To append shapes to a page with D3, you need to have an SVG canvas element somewhere in your page's DOM. You could either add this SVG canvas to the page as you write the HTML, or you could append it using the D3 syntax you've learned:

```
d3.select("body").append("svg");
```

After we have an SVG canvas on our page, we can append various shapes to it using the same select() and append() syntax we've been using for <div> elements.

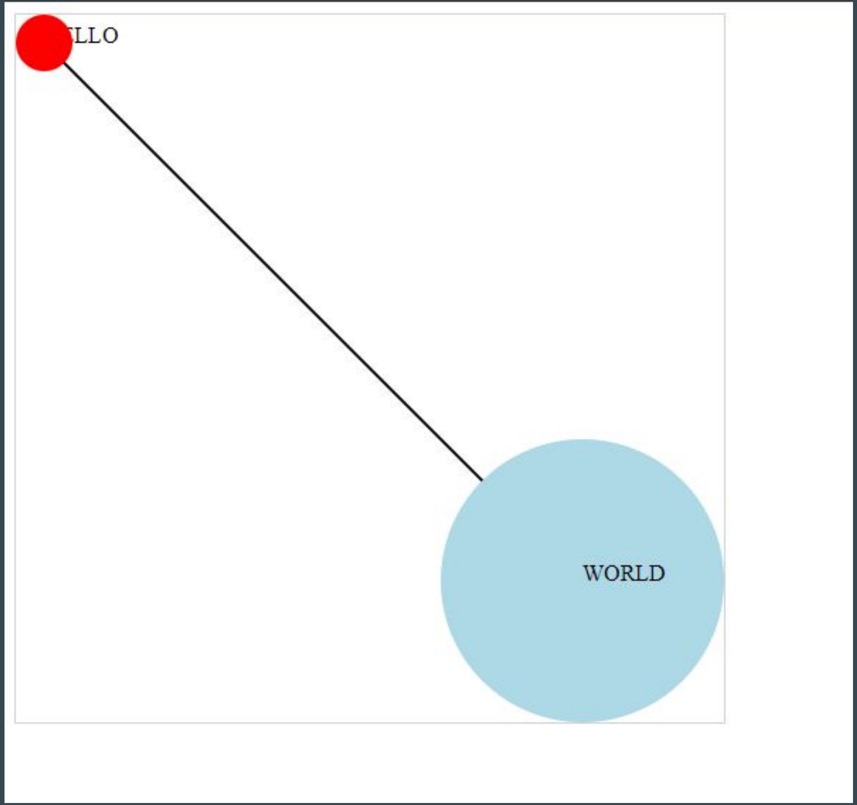
Listing 1.9 A simple web page with an SVG canvas (D3IA)



```html
<html>
<head>
    <script src="d3.v4.min.js"></script>
</head>
<body>
    <div id="vizcontainer">
        <svg style="width:500px;height:500px;border:1px lightgray solid;" />
    </div>
</body>
</html>
```

Creating lines and circles with select and append - JSFIDDLE

Notice that your circles are drawn over the line, and the text is drawn above or below the circle, depending on the order in which you run your commands, as you can see.

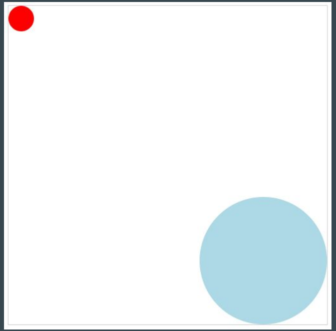


A conversation with D3

Writing Hello World with languages is such a common example that I thought we should give the world a chance to respond.

Let’s add the same big circle and little circle from before, but this time, when we add text, we’ll include the .style (opacity) setting that makes our text invisible, as shown in the following listing. We’ll also give each text element a .attr(id) setting so that the text near the small circle has an id attribute with the value of a, and the text near the large circle has an id attribute with the value of b.

Two circles, no line, and no text - JSFIDDLE



Now you make the text appear using the .transition() method with the .delay() Add:

d3.select("#a").transition().delay(1000).style("opacity", 1);
d3.select("#b").transition().delay(3000).style("opacity", .75);

Congratulations! You've made your first dynamic data visualization. The .transition() method indicates that you don't want your change to be instantaneous. By chaining it with the .delay() method, you indicate how many milliseconds to wait before implementing the style or attribute changes that appear in the chain after that .delay() setting.

Let’s look at another .transition() setting - JSFIDDLE

You can set a .delay() before applying the new style or attribute, but you can also set a .duration() over which the following change is applied. What will the result be?

d3.selectAll("circle").transition().duration(2000).attr("cy", 200);

The .duration() method, as you can see, adjusts the setting over the course of the amount of time (again, in milliseconds) that you set it for. That covers the basics of how D3 works and how it’s designed, and these fundamental
concepts will surface again and again throughout the following chapters, where you’ll learn more complicated variations on representing and manipulating data.

This weeks lab

Week 2 Lab

References

All content is taken verbatim from Chapter 1/2 of D3.js in Action