See the companion code running at: https://pure-hamlet-59256.herokuapp.com/

Visualisation with JavaScript

An introduction for designer/developers

NOTE: THIS SLIDE DECK IS **ORGANISED BY WEEK** (WITH EACH WEEK HAVING A BLUE TITLE PAGE), AND ARRANGED IN **REVERSE CHRONOLOGICAL ORDER** (SO THE MOST RECENT WEEK WILL BE FIRST). HOPE THAT MAKES SENSE 😀

The code is now running at: https://pure-hamlet-59256.herokuapp.com/

Week 9. Changes and transitions

The code is now running at: https://pure-hamlet-59256.herokuapp.com/

Week 8.
The General Update Pattern

The code is now running at: https://pure-hamlet-59256.herokuapp.com/

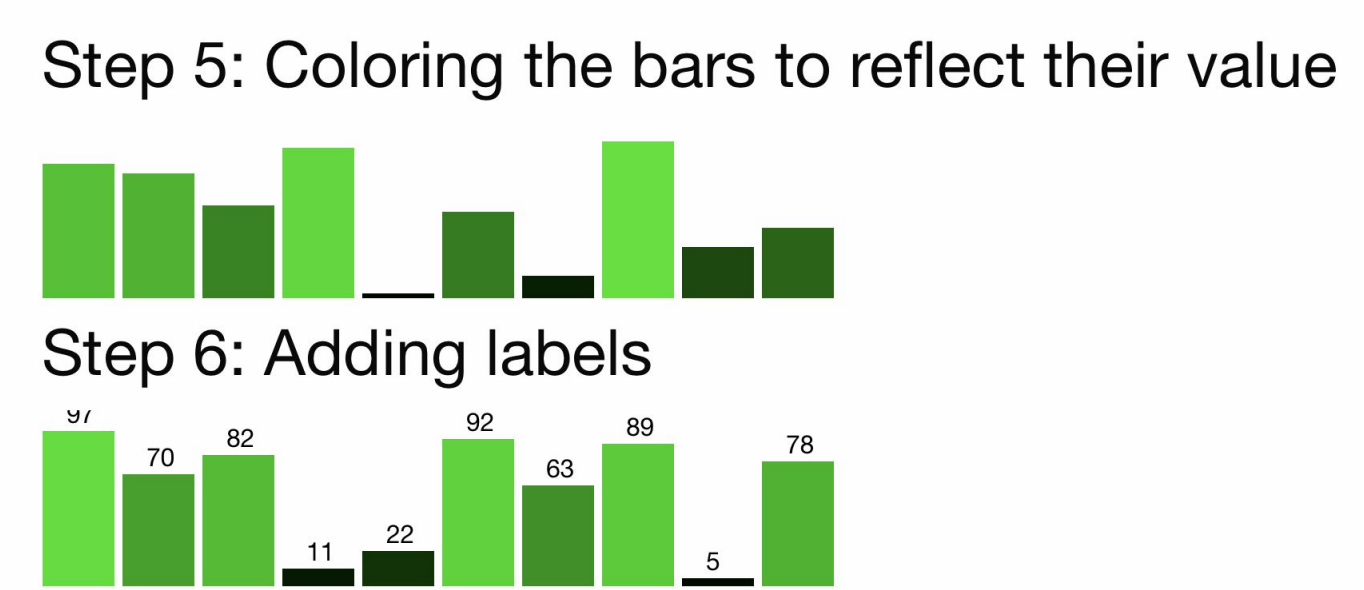
Week 7.
Introducing Axes

Week 6. Continuing with scales, including scaleTime()

Week 5. Completing our chart and looking at D3 scales

Adding dynamic colour and labels to the chart


See building-a-real-chart-in-d3.html



Scales in D3

“D3 scales are functions that map from an **input domain** to an **output range**”

 a close look at D3 scales is provided in d3-scales.html

 See building-a-real-chart-in-d3.html for the full process, beginning to end

Week 4. Step-by-step to a data-driven, chart in D3

The data join: selecting things that don't exist

```
const svg = d3
    .select('.step-one')
    .append('svg')
    .attr('width', 500)
    .attr('height', 300);

svg.selectAll("circle")
    .data(dataset)
    .enter()
    .append("circle")
    .attr("cx", function (d) {
        return d.x;
    })
    .attr("cy", function (d) {
        return d.y;
    })
    .attr("r", 2.5);
```

Thinking with joins

"This code does exactly what you need: it creates a circle element for each data point, using the x and y data properties for positioning. But what's with the `selectAll("circle")`? **Why do you have to select elements that you know don't exist in order to create new ones?** WAT.

Here's the deal. Instead of telling D3 *how* to do something, tell D3 *what* you want. You want the `circle` elements to correspond to data. You want one circle per datum. Instead of instructing D3 to create circles, then, tell D3 that the selection "circle" should correspond to data. **This concept is called the *data join*"**

https://bost.ocks.org/mike/join/

Week 3
The data join - and a mixed bag

But first...

The steps for all visualisations

This is the sequence you need to follow

1. **Prepare** your data
   a. Clean it (strings to ints etc.)
   b. Shape it (rollup etc.)
2. **Bind** data to DOM elements
3. **Draw** the visualisation
4. Add **interaction**
5. Add **animations**

🤦🏾‍♀️ See:

- our-first-react-component.html
- our-first-react-component-with-d3.html

How might React fit into this picture? Do we really need it?

Let’s look at the repository and discuss this. We might not need it. Is there an easier/better way to link our range slider to the data? Could we have all the code in a vanilla event listener?

Let’s look at an example Treemap with updating data

There’s an example in the repository.

 See:

- Using-someone-elses-visualisation.html
- updating-visualisation-data.html

Week 2:
D3 Modules and methods;
Data cleaning and prep;
Code ‘borrowing’

🤦🏾‍♀️ there’s one in week-one-challenge-a-solution.html

But first… let’s begin with a review of solutions to Week 1 challenge

D3 provides a lot, but it's all very well documented

- D3 has **30 modules** (d3-shape, d3-scale etc.) that together provide around **900 functions**
- Because **D3 is modular**, you need only load those modules you need.
- All modules are documented, along with their dependencies at https://github.com/d3
    - For example, here's the documentation for d3-fetch https://github.com/d3/d3-fetch

👩‍💻 follow along using-rollup-to-shape-data.html

Fetching data with d3-fetch

As an example of using a d3 module

http://localhost:8080/loading-csv-with-d3.html

Let's look at the d3-fetch module and some of its methods. Here we see:

- **Loading only the required D3 modules** (d3-fetch and d3-dsv, which is a dependency)
- Using **d3-csv** and **d3-json** to fetch and parse data from the filesystem or API
  - There are other methods that can fetch all sorts of other things like image or binary data.

👩🏾‍💻 follow along using loading-data-with-d3.html

D3 data conversion - Part 1

‘cleaning’ our data

- Use the **type** function as the ***second argument*** to d3.json to ‘pipe’ your data through it. Think of this as being similar to JavaScript Array.prototype.map()

👩🏽‍💻 follow along using-rollup-to-shape-data.html

Data conversion - Part 2

Shaping the data for visualisation

- D3 visualisations will typically need data that's in a particular shape.
- Let's see how we can use **d3-rollup()** to transform some clean data into a usable shape
  - **Remember:** d3-rollup() returns a Map, but - for binding data to the DOM - d3 tends to prefer Arrays of Objects (which is why we're doing the transform)

A quick aside, let's look at using someone else's D3 visualisations

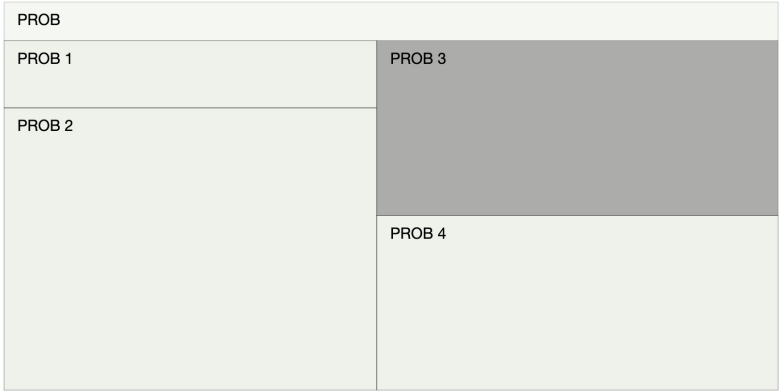
This is jumping ahead a bit....

...but, given what we've got ahead of us this sprint, I thought it might be useful.

Here's something I've 'hacked' together by manipulating someone else's visualisation.

Here's the original:
https://bl.ocks.org/d3indepth/a6ca05860b7249ebe163a212a4abd9cf

Here's what I came up with:
http://localhost:8080/using-someone-elses-visualisation.html



Let's go through the code together to work out what it's doing...

Week 1: Introducing D3 and some basics of SVG

What D3 *is not*

A strange place to start, I know - but there are a lot of misconceptions

- **It is not a charting library** (though many JavaScript charting libraries use D3 ‘under the hood’)
- **It has nothing to do with ‘3D’** - use three.js for that
- **It does not draw shapes itself.** Instead it leaves that to a ‘render technology’ (typically SVG, but can be Canvas or even HTML)
    - Think of D3 as the artist and SVG as the medium

What D3 *is*

In a nutshell, it's Data Driven Documents

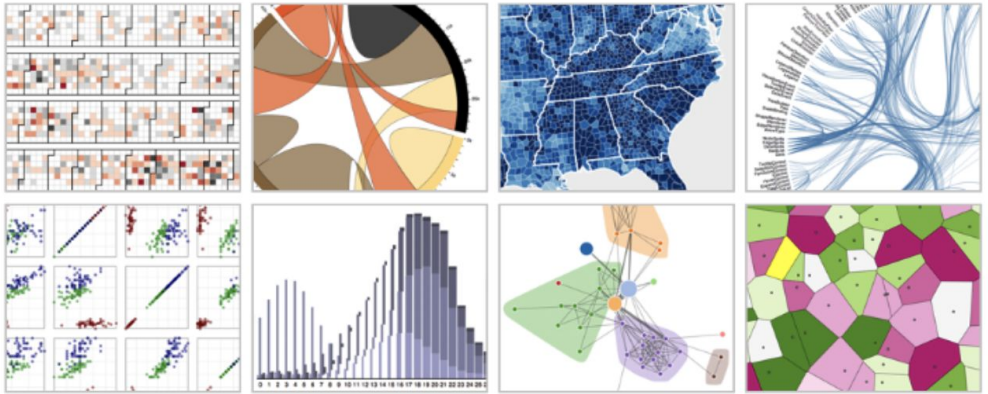
- Started in 2011 at the Stanford Visualisation Group (See the paper here: http://vis.stanford.edu/papers/d3)
- D3 is *Data Driven Documents* (D3):
  - Binds data to a DOM
  - Enables transitions in the DOM to represent changes in the data
- A modular system (since version 4)

The Stanford paper

*"...selectively **bind** input data to arbitrary **document elements**, applying dynamic transforms to both **generate and modify content**"*

D3: Data-Driven Documents

Michael Bostock, Vadim Ogievetsky, Jeffrey Heer



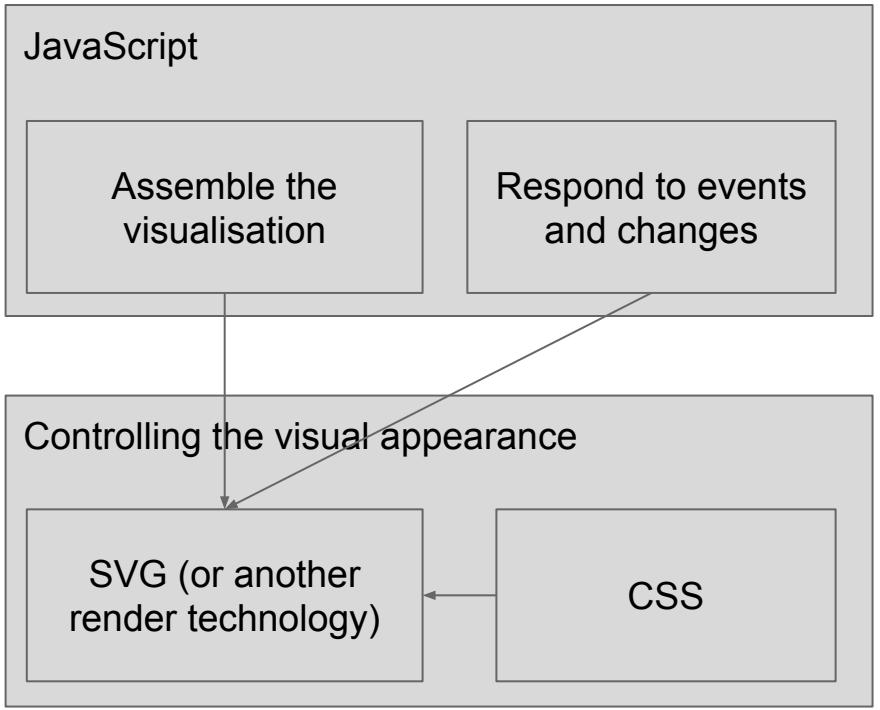
Interactive visualizations built with D3. From left to right: calendar view, chord diagram, choropleth map, hierarchical edge bundling, scatterplot matrix, grouped & stacked bars, force-directed graph clusters, Voronoi tessellation.

ABSTRACT

Data-Driven Documents (D3) is a novel representation-transparent approach to visualization for the web. Rather than hide the underlying scenegraph within a toolkit-specific abstraction, D3 enables direct inspection and manipulation of a native representation: the standard document object model (DOM). With D3, designers selectively bind input data to arbitrary document elements, applying dynamic transforms to both generate and modify content. We show how representational transparency improves expressiveness and better integrates with developer tools than prior approaches, while offering comparable notational efficiency and retaining powerful declarative components. Immediate evaluation of operators further simplifies debugging and allows iterative development. Additionally, we demonstrate how D3 transforms naturally enable animation and interaction with dramatic performance improvements over intermediate representations.

Technologies involved

- You select an **HTML** element where you will place your D3 visualisation
- You can use **CSS to style elements** of your visualisation (you can also control some of this by amending the properties of the SVG shapes)
- **JavaScript** does two things:
  - ‘assembles’ the visualisation from the data
  - Allows you to control what happens in response to user actions
- **SVG** (or Canvas - or even HTML) draws the elements of your visualisation
  - Note: you don’t tend to write much SVG by hand when working with D3



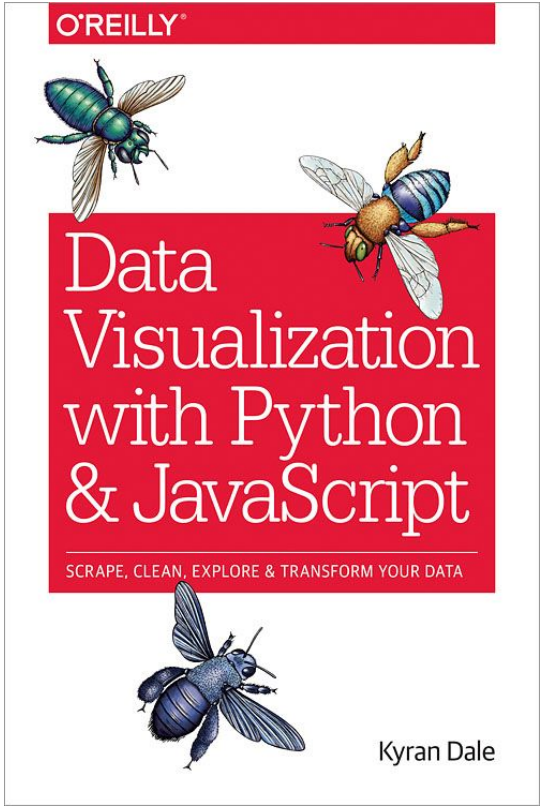
But why are you recommending D3 and not just a charting library?

Some benefits of D3

- Some of our ideation outputs are highly bespoke and complex. With D3 we'll be able to control these (without relying on hobbling together 3rd party 'pre-packaged' visualisations that are unlikely to meet development standards (esp. a11y)
- D3 is the leader (and behind many charting libraries)

“Dataviz on the Web is an exciting place to be right now with innovations in interactive visualizations coming thick and fast, and many (if not most) of them being developed with D3”

Kyran Dale



A quick SVG refresher

What SVG is

The helicopter view

- XML based language for describing two-dimensional graphics - *what HTML is to text, SVG is to graphics*
- A web standard 😍
- Can be used in HTML and CSS (either inline or referencing a file)*
- Easy to get to grips with but can be quite a deep topic when you look at advanced topics

---

** As always, check browser support*

Some common SVG elements

...there are many more - see https://developer.mozilla.org/en-US/docs/Web/SVG/Element

- `<g>` - the group element
- `<line>`
- `<circle>`
- `<text>`
- `<path>`
- `<rect>`
- `<a>`
- `<ellipse>`
- `<path>`
- `<polygon>`

Some common SVG attributes

...there are many more - see https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute

- `fill`
- `stroke`
- `tabindex`
- `cy`, `cx`, `r` - for circle
- `x`, `y`, `width`, `height` – for rect
- `x1`, `x2`, `y1`, `y2`, `stroke` – for line

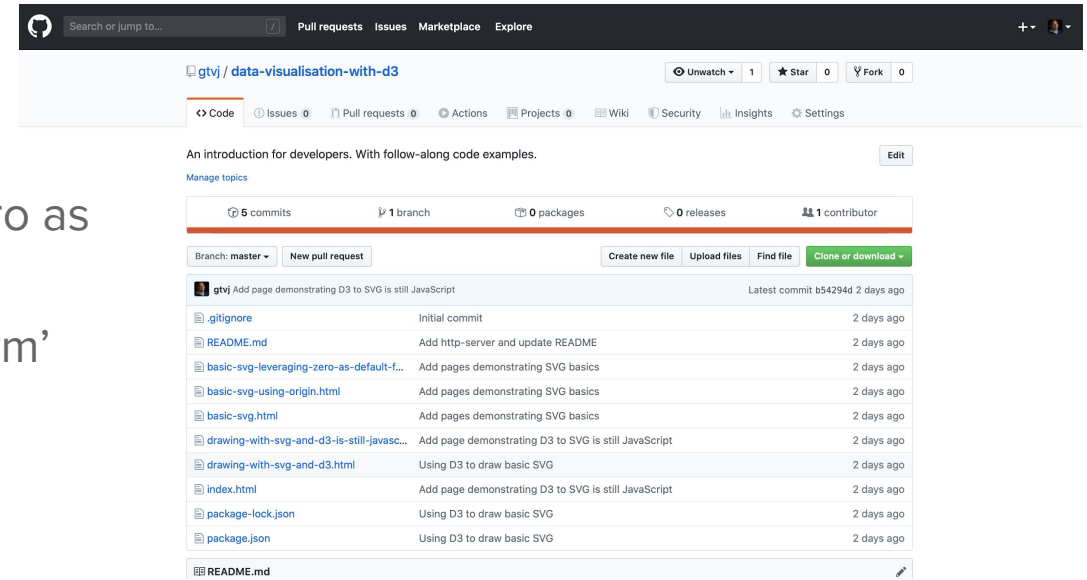
Week 1: Code demo/challenge

Repository

https://github.com/gtvj/data-visualisation-javascript

Let's look at the repository:

- A basic SVG
- A basic SVG leveraging zero as default for attributes
- A basic SVG using 'transform' to set origin on a group
- Drawing an SVG with D3
- D3 is still JavaScript



Challenge

Using the code we've looked at (especially the last example) use JavaScript and D3 to create a visualisation that shows 50 circles, of random size, color and position on the SVG.