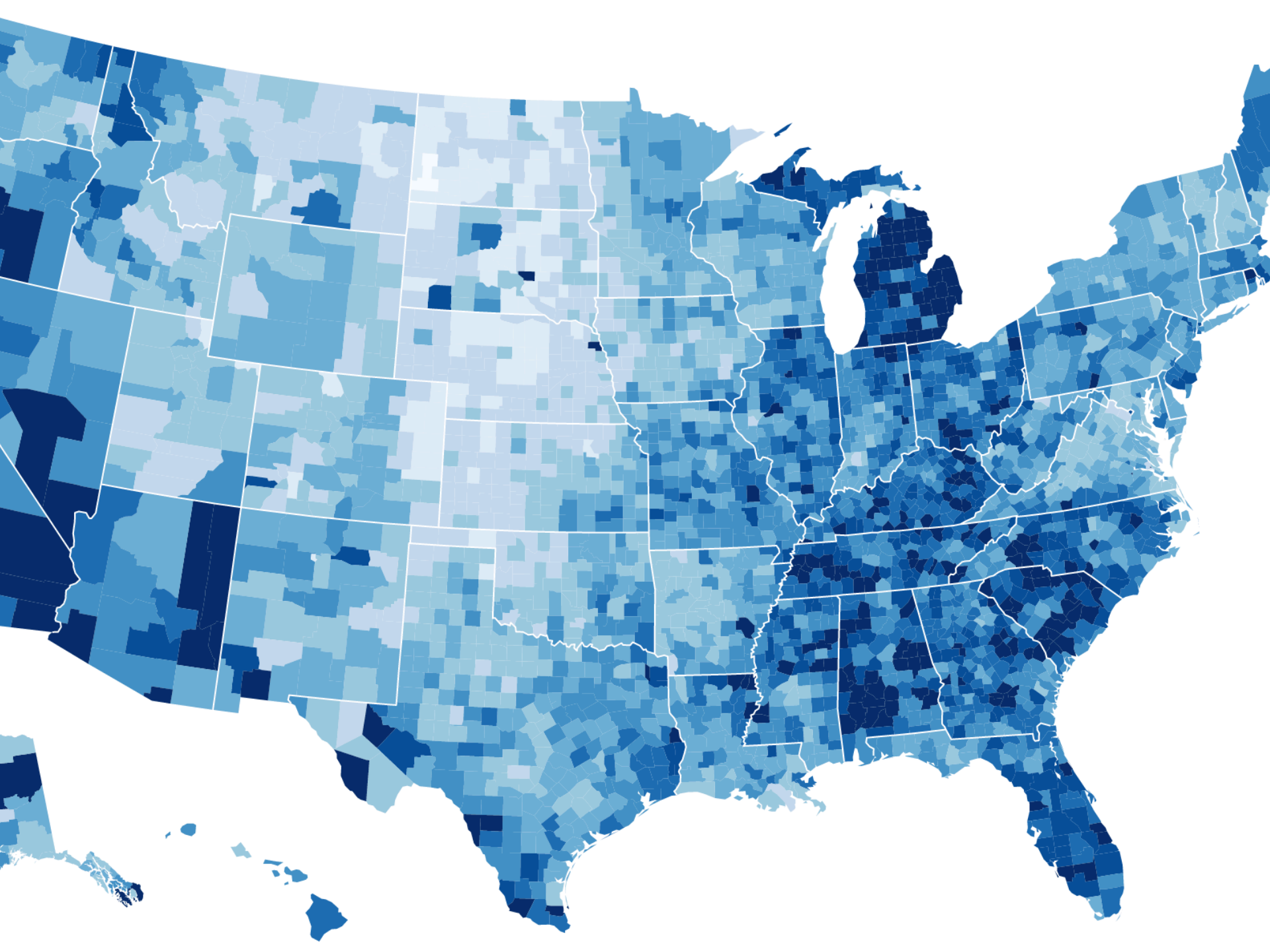
D3.js Workshop

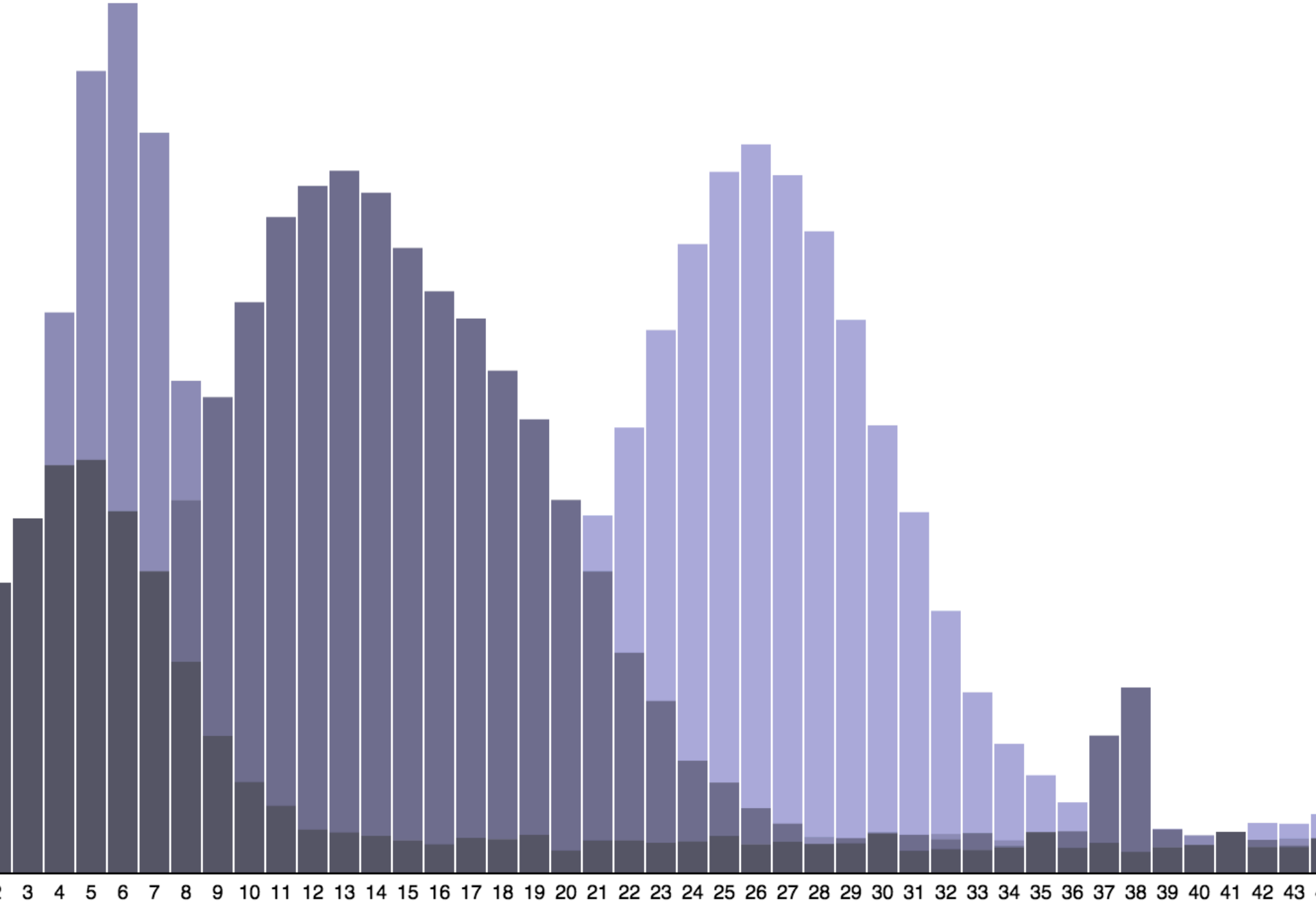
EMILY FUHRMAN — @xxzvx

https://github.com/emilyfuhrman/map-club/tree/master/2016_Fall/Session_08/

What is D3?

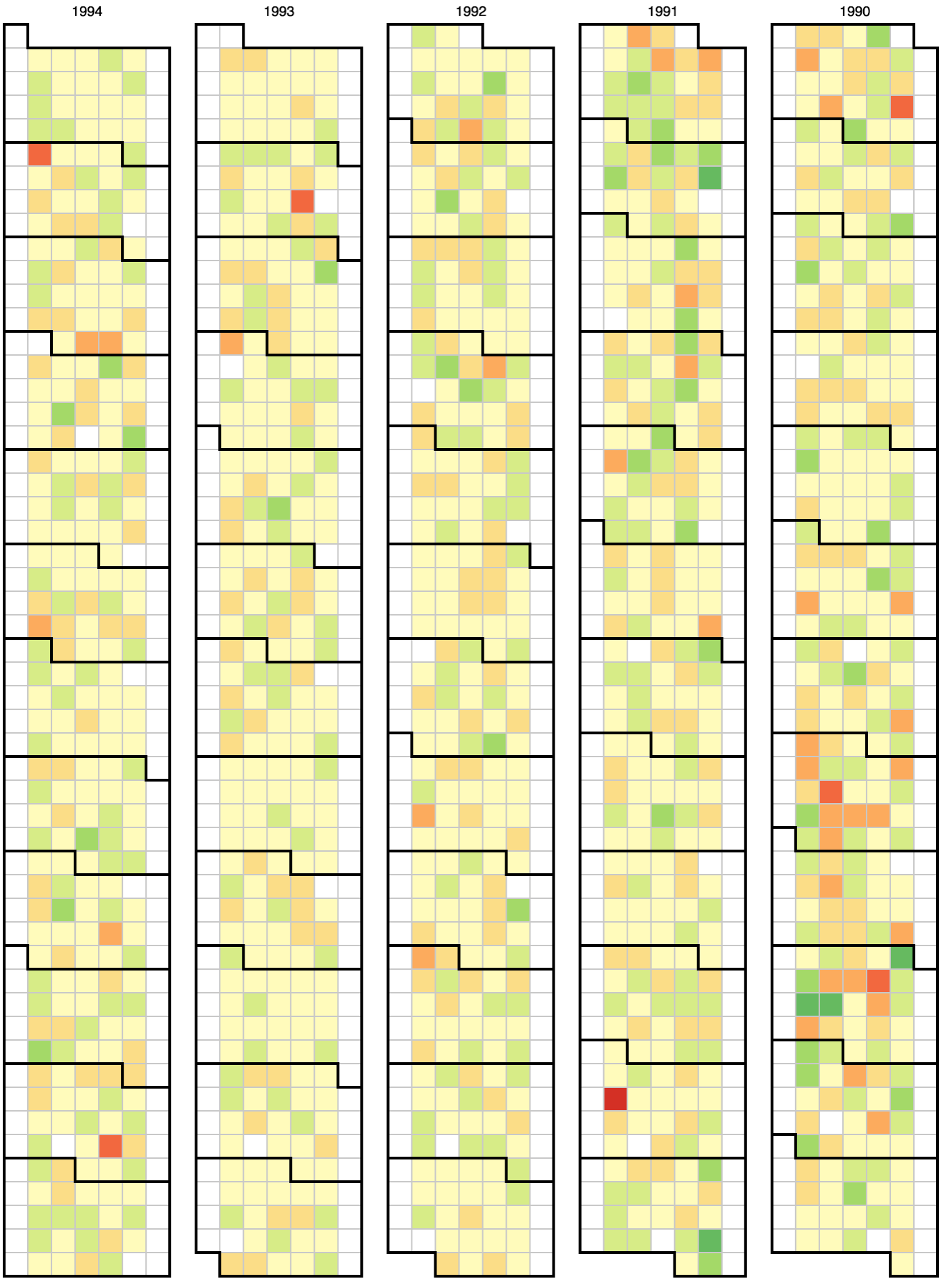
A general-purpose visualization library for HTML and SVG.

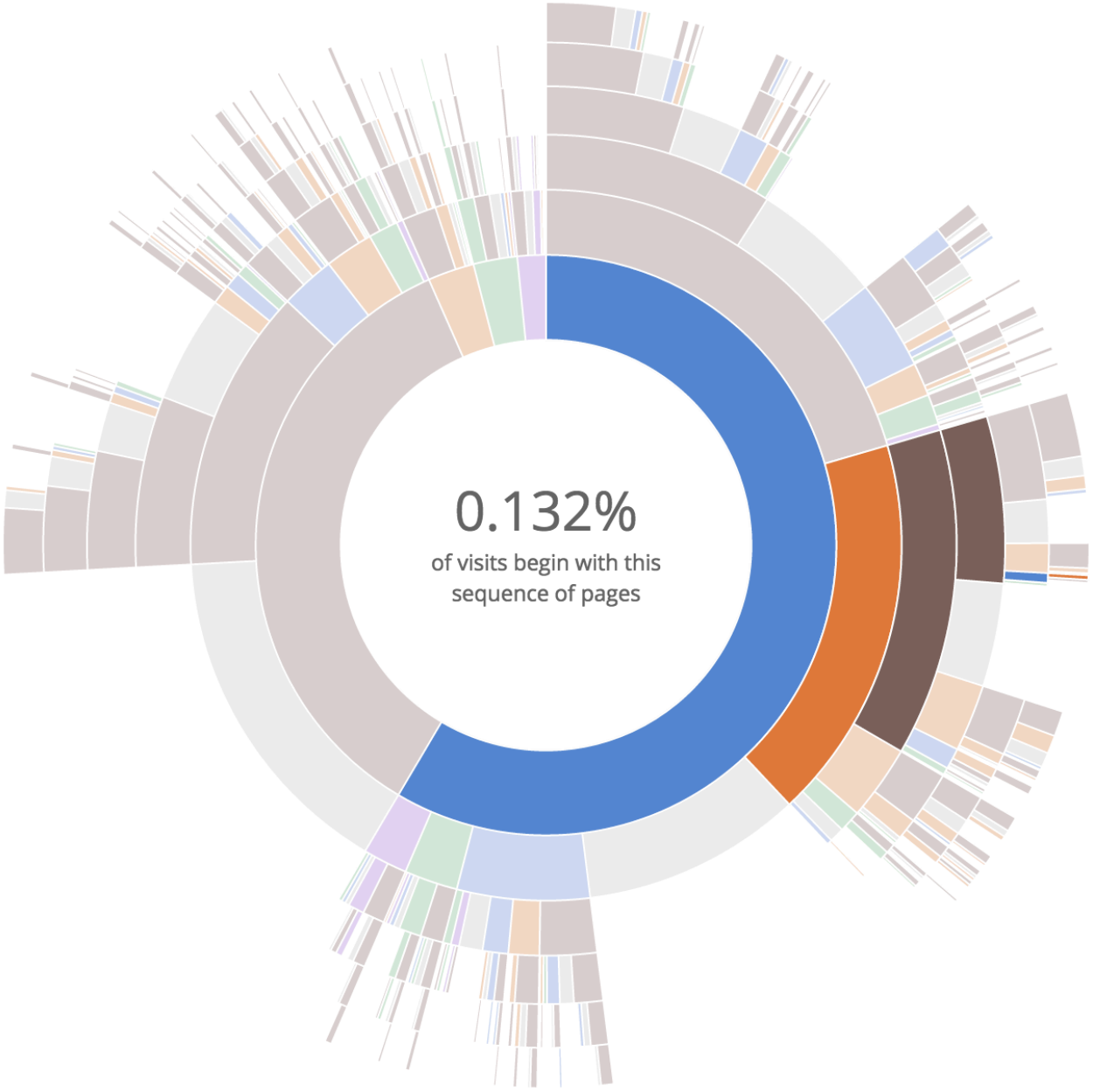


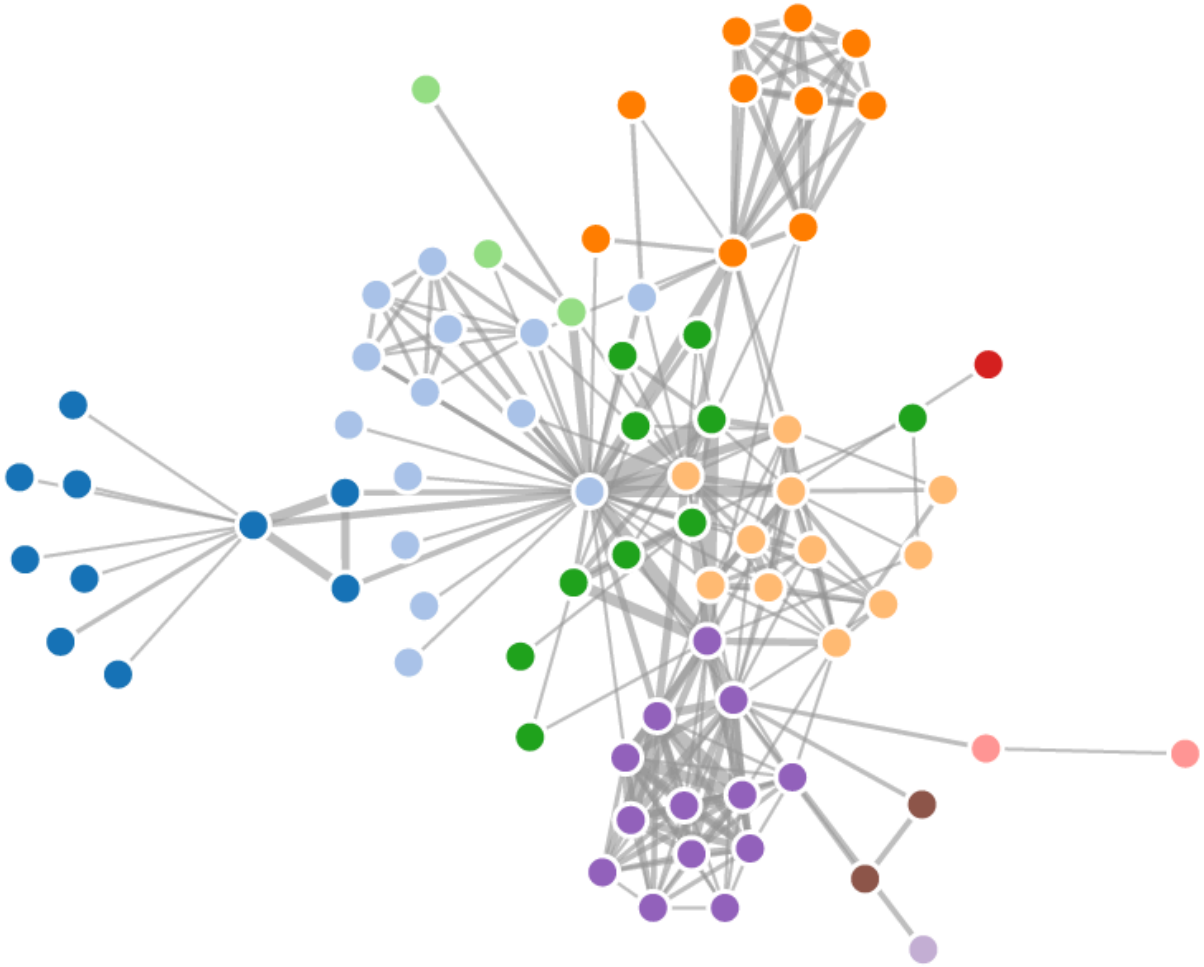

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43

Efficiently transform data into elements in a browser.

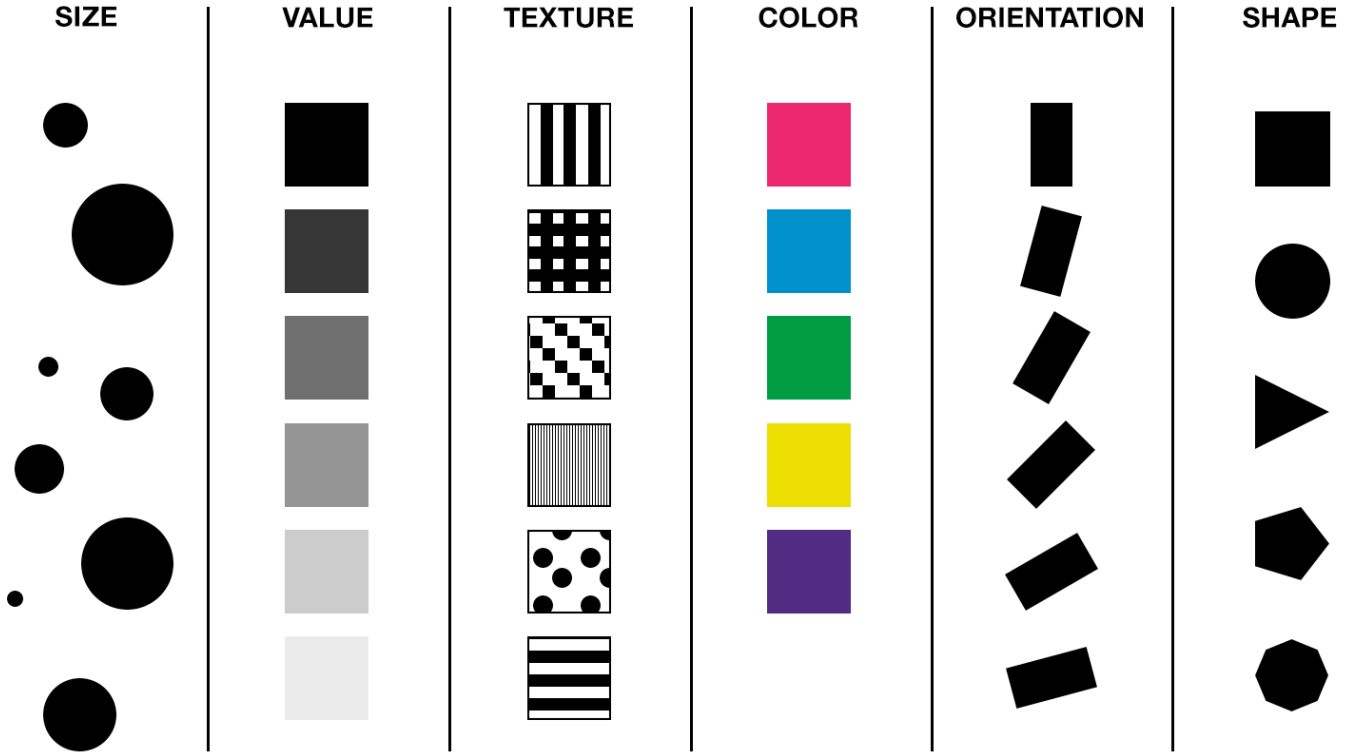
Visualizations are comprised of basic graphical forms.


1994
1993
1992
1991
1990


0.132%
of visits begin with this
sequence of pages



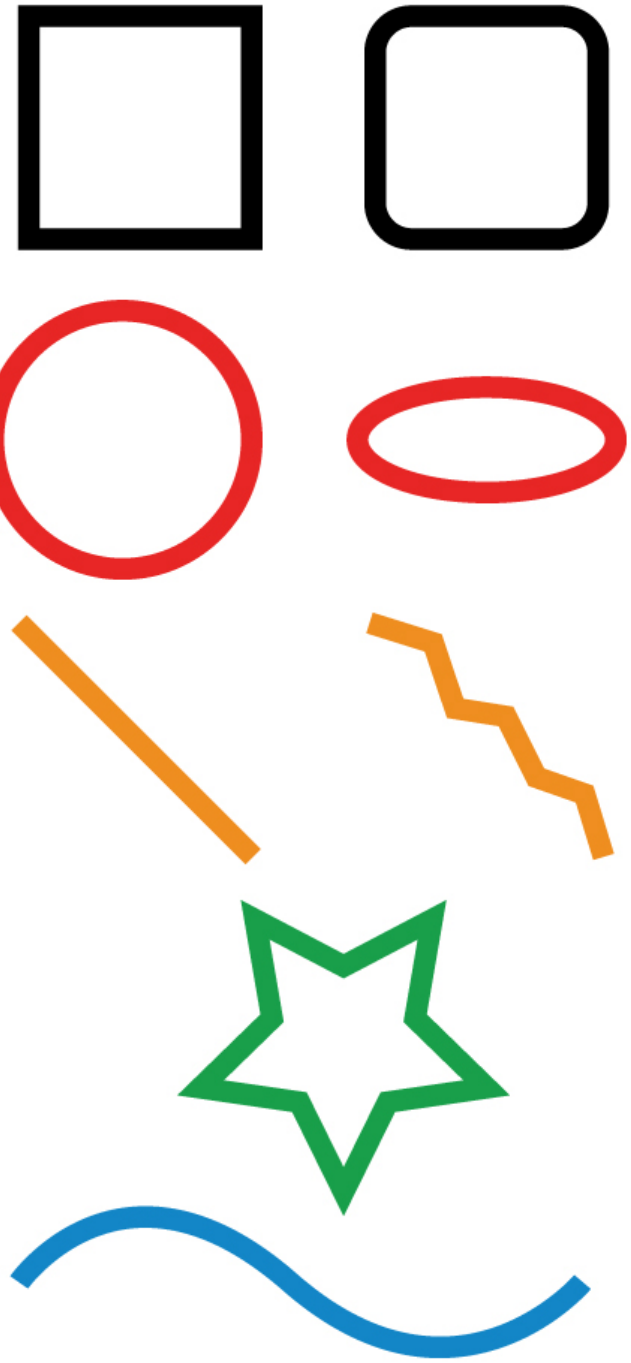
...with attributes.



Jacques Bertin: six retinal properties

SVG:
DOM for graphics.

Rectangle
Circle
Ellipse
Line
Polyline
Polygon
Path



JS data types & JSON.

String
Number
Boolean
Array
Object

STRING

A series of characters enclosed in double- or single-quotes.

```
var x = "hello”
var x = "34"
```

NUMBER

Integer or float.

```
var x = 34
var x = 34.00
```

BOOLEAN

True or false.

```
var x = true
var x = false
```

ARRAY

Square brackets. Items separated by commas.

```
var x = [1, 2, 3]
var x = ["this", "that"]
```

OBJECT

Curly brackets. Properties are written as name/value pairs, separated by commas.

```
var x = {
  "name":"bob",
  "age":35,
  "location":"NY"
}
```

JSON is a syntax for data storage and exchange.

JSON syntax is derived from JavaScript syntax.

JSON data is written in name/value pairs.

```
"city":"new york"
```

JSON objects are stored inside curly brackets...

```
{"city":"new york"}
```

...which can contain multiple name/ value pairs.

```
{"city":"new york", "state":"NY"}
```

JSON arrays are written inside square brackets.

```
"locations":[
   {"city":"new york", ”state":"NY"},
   {"city":"los angeles", ”state":"CA"},
   {"city":"chicago", ”state":"IL"}
];
```

To work with JSON in JS, create an array and assign data to it.

```
var locations = [
   {"city":"new york", ”state":"NY"},
   {"city":"los angeles", ”state":"CA"},
   {"city":"chicago", ”state":"IL"}
];
```

In Excel, that data would look like this.

	A	B	C
1	**city**	**state**	
2	new york	NY	
3	los angeles	CA	
4	chicago	IL	
5			
6			
7			
8			
9			

Elements in an array can be accessed using their position (index).

```
locations[0]
{"city":"new york", ”state":"NY"}

locations[2]
{"city":"chicago", ”state":"IL”}
```

D3: a closer look

D3 magic: THE JOIN

- Pairs a data object with an element
- Keeps track of new and old objects
- Lets you animate differences between new and old

Pie chart example:

http://bl.ocks.org/dbuezas/9572040

enter, update, exit

Start with a selection.

```
//this is an empty selection
//looks for instantiations of data

var elements = d3.selectAll('div')
```

Join selected elements with data items.

```
var elements = d3.selectAll('div')
  .data([1,2,3]);
```

enter()

For every part of the data that does not correspond to an existing element, add an element.

```
elements.enter().append('div');
```

(update)

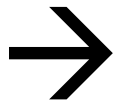
For each element in the selection, update attributes.

```
elements
  .attr('background','red');
```

exit()

Remove elements that no longer correspond to the data.

```
elements.exit().remove();
```

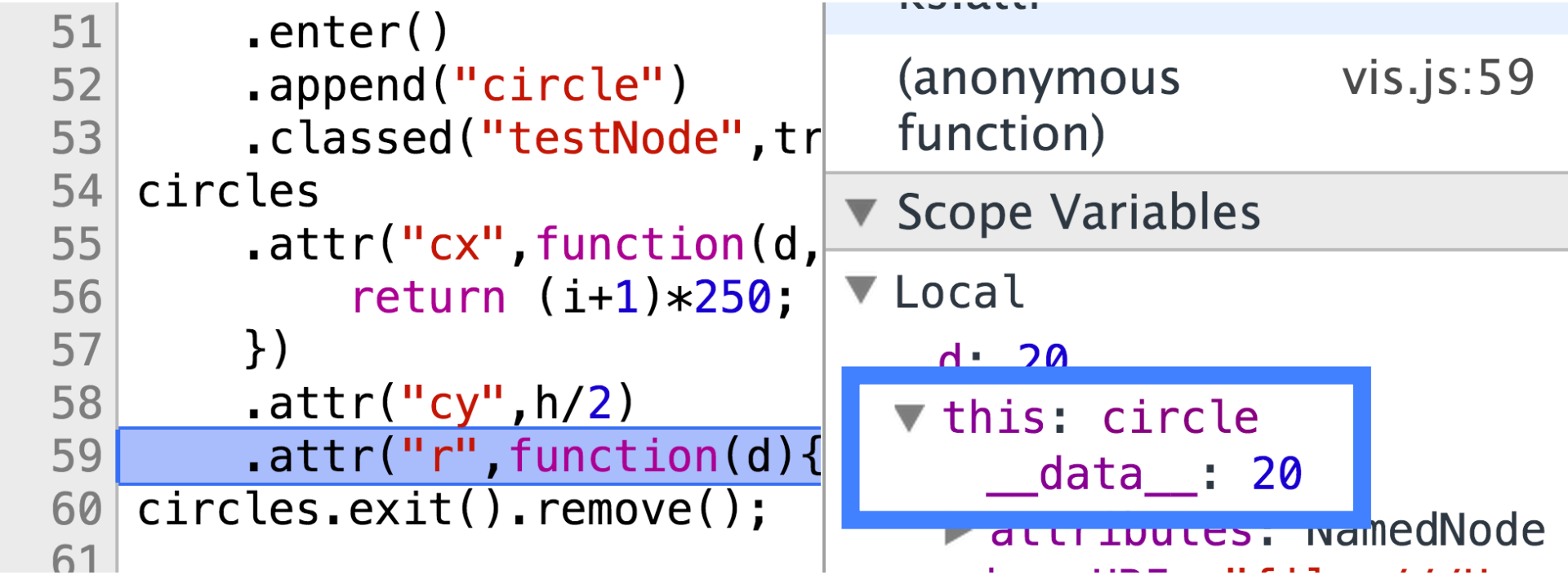


data bound to the DOM.

Each element has a property that stores the data.

The __data__ property.

```
46
47  var circles = svg                          //select the ele
48      .selectAll("circle.testNode")          //here we select
49      .data(sampleData);                     //we want the sa
50  circles
51      .enter()
52      .append("circle")                              //append
53      .classed("testNode",true);                     //assign
54  circles
55      .attr("cx",function(d,i){                      //"cx" i
56          return (i+1)*250;                          //for ea
57      })
58      .attr("cy",h/2)                                //"cy" i
59      .attr("r",function(d){debugger; return d; });
60  circles.exit().remove();
61
```


51 .enter()
52 .append("circle")
53 .classed("testNode",tr
54 circles
55 .attr("cx",function(d,
56 return (i+1)*250;
57 })
58 .attr("cy",h/2)
59 .attr("r",function(d){
60 circles.exit().remove();
61
(anonymous function) vis.js:59
Scope Variables
Local
this: circle
__data__: 20

```
// d3 has a few different
// functions that set stuff

.text()
.property()
.style()
.attr()
```

```
// each takes a function

.attr('foo', function() { })
```

```
// and that function gets data
// from your .data()

.attr('foo', function(d) {
  return d.foo;
})
```

You can chain these, as well as use them to pull an existing value.

MORE D3 FEATURES:

Scale
Projections

LET'S MAKE SOME CIRCLES!!

Setup

```
//append an SVG element
//to the div with ID “vis”

var svg = d3.select("#vis")
   .append("svg")
```

```
//set the width and height

var svg = d3.select("#vis")
   .append("svg")
   .attr("width",w)
   .attr("height",h);
```

SVG circles have a center point position and a radius.

`(cx, cy, r)`

```
//full code

var circles = svg.selectAll('circle')
   .data([50,100,150])
   .enter()
   .append('circle')
   .attr('cx',500)
   .attr('cy',300)
   .attr('r',function(d){
      return d;
   });
```

```
//create an empty selection
//this looks for instantiations of data

var circles = svg.selectAll('circle')
```

```
//this is data, which
//would be bound to a selection

var circles = svg.selectAll('circle')
   .data([50,100,150])
```

```
//ENTER: for every time we see data,
//but do not see a corresponding element

var circles = svg.selectAll('circle')
   .data([50,100,150])
   .enter()
```

```
//append an element

var circles = svg.selectAll('circle')
   .data([50,100,150])
   .enter()
   .append('circle')
```

```
//set x-position and y-position

var circles = svg.selectAll('circle')
   .data([50,100,150])
   .enter()
   .append('circle')
   .attr('cx',500)
   .attr('cy',300)
```

```
//finally, set circle radius
//based on value from the array

var circles = svg.selectAll('circle')
   .data([50,100,150])
   .enter()
   .append('circle')
   .attr('cx',500)
   .attr('cy',300)
   .attr('r',function(d){
      return d;
   });
```

```
//full code

var circles = svg.selectAll('circle')
   .data([50,100,150])
   .enter()
   .append('circle')
   .attr('cx',500)
   .attr('cy',300)
   .attr('r',function(d){
      return d;
   });
```

Only one?

```
//add a function to ‘cx’

var circles = svg.selectAll('circle')
   .data([50,100,150])
   .enter()
   .append('circle')
   .attr('cx',function(d,i){
      return (i*300);
   })
   .attr('cy',300)
   .attr('r',function(d){
      return d;
   });
```

```
//tweak it

var circles = svg.selectAll('circle')
   .data([50,100,150])
   .enter()
   .append('circle')
   .attr('cx',function(d,i){
      return (i*300) + 250;
   })
   .attr('cy',300)
   .attr('r',function(d){
      return d;
   });
```

Some challenges.

- Give the circles a red stroke.
- Make the data drive the stroke-width of the circles.
- Make the circles rectangles.

A good template.

selection

enter()

attributes

interaction

exit()

selection

```
var circles = svg.selectAll('circle')
   .data([50,100,150]);
```

enter()

```
circles.enter().append('circle');
```

attributes

```
circles
   .attr('cx',500)
   .attr('cy',300)
   .attr('r',function(d){
      return d;
   });
```

interaction

```
circles
   .on('mouseover',function(){});
```

exit()

```
circles.exit().remove();
```

```
var circles = svg.selectAll('circle')
   .data([50,100,150]);
```

```
circles.enter().append('circle');
```

```
circles
   .attr('cx',500)
   .attr('cy',500)
   .attr('r',function(d){
      return d;
   });
```

```
circles
   .on('mouseover',function(){})
```

```
circles.exit().remove();
```

Challenges.

Challenge: give the circles a red stroke.

```
//red stroke code

var circles = svg.selectAll('circle')
   .data([50,100,150])
   .enter()
   .append('circle')
   .attr('cx',function(d,i){
      return (i*300) + 250;
   })
   .attr('cy',300)
   .attr('r',function(d){
      return d;
   })
   .style('stroke’,'red’);
```

Challenge: make the data drive the stroke-width of the circles.

```
//stroke-width code

var circles = svg.selectAll('circle')
   .data([50,100,150])
   .enter()
   .append('circle')
   .attr('cx',function(d,i){
      return (i*300) + 250;
   })
   .attr('cy',300)
   .attr('r',function(d){
      return d;
   })
   .style('stroke’,'red')
   .style('stroke-width',function(d,i){
      return (i*2);
   });
```

Challenge: make the circles rectangles

SVG rectangles have an x-position, a y-position, width, and height.

(x, y, width, height)

```
//rectangle code

var rects = svg.selectAll('rect')
   .data([50,100,150])
   .enter()
   .append('rect')
   .attr('x',function(d,i){
      return (i*300) + 250;
   })
   .attr(‘y',300)
   .attr('width',function(d){
      return d;
   })
   .attr('height',function(d){
      return d*20;
   });
```

Resources.

SVG Basic Shapes
https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial/Basic_Shapes

JSON Syntax
http://www.w3schools.com/json/json_syntax.asp

D3 Joins
http://bost.ocks.org/mike/join/
http://bost.ocks.org/mike/circles/

People who write about D3
http://macwright.org
http://jasondavies.com
http://mbostock.github.com

Thank you!

EMILY FUHRMAN — @xxzvx