D3 Introduction

Slides adapted from

Maneesh Agrawala

Jessica Hullman

CS 448B: Visualization
Spring 2016

Topics

Motivation

DOM Manipulation

Scales
Axes
Coordinate system
Path generators
Layouts

[Adapted from Mike Bostock's D3 Workshop]

Motivation

Visualization with Web Standards

Built on top of HTML, CSS, JavaScript, and SVG
Data → DOM Elements
D3: Data-Driven Documents

Benefits:
Expressivity
Developing and debugging tools
Documentation, resources, community

hello-world.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
</head>

<body>
  Hello, world!
</body>
</html>
```

hello-css.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <style>
    body { background: steelblue; }
  </style>
</head>

<body>
  Hello, world!
</body>
</html>
```

hello-svg.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <style> /* CSS */ </style>
</head>

<body>
  <svg width="960" height="500">
    <text x="10" y="50">
      Hello, world!
    </text>
  </svg>
</body>
</html>
```

hello-javascript.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <style> /* CSS */ </style>
</head>

<body>
  <svg width="960" height="500">
    <text x="10" y="50">
      Hello, world!
    </text>
  </svg>
  <script>
    console.log("Hello, world!");
  </script>
</body>
</html>
```

hello-d3.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <style> /* CSS */ </style>
</head>

<body>
  <script src="https://d3js.org/d3.v3.min.js"></script>
  <script>
    // JavaScript
  </script>
</body>
</html>
```

Local development

Run on a local server

```
> cd path/to/your/project
> python -m SimpleHTTPServer
```

http://localhost:<port>

DOM Manipulation

Selection

```
// select all SVG circle elements
var circles = d3.selectAll("circle");


// set attributes and styles
circles.attr("cx", 20);
circles.attr("cy", 12);
circles.attr("r", 24);
circles.style("fill", "red");


// method chaining
d3.selectAll("circle")
      .attr("cx", 20)
      .attr("cy", 12)
      .attr("r", 24)
      .style("fill", "red");
```

```
<html>
…
<svg>
  <circle cx="10" cy="10" r="5" />
  <circle cx="20" cy="15" r="5" />
</svg>
```

Selection

```
var rect = d3.selectAll(“rect”)
      .attr(“x", 20)
      .attr(“y", 12)
      .attr(“width", 24)
      .attr(”height", 24);

var line = d3.selectAll(“line”)
      .attr(“x1", 20)
      .attr(“y1", 12)
      .attr(“x2", 40)
      .attr(”y2", 24);

var text = d3.selectAll(“text”)
      .attr(“x", 20)
      .attr(“y", 12)
      .text(“Hello!”);

var div = d3.selectAll(“div”)
      .text(“Hello!”);
```

```
var firstRect = d3.select(“rect”)
      .attr(“x", 20)
      .attr(“y", 12)
      .attr(“width", 24)
      .attr(”height", 24);
```

Selection.append

```
// select the <body> element
var body = d3.select("body");

// add an <h1> element
var h1 = body.append("h1");
h1.text("Hello!");
```

Selects one element, adds one element.

Selection.append

```
// select the <body> element
var sections = d3.selectAll(”section");

// add an <h1> element
var h1 = sections.append("h1");
h1.text("Hello!");
```

Selects multiple elements, adds one element to each.

Data → elements

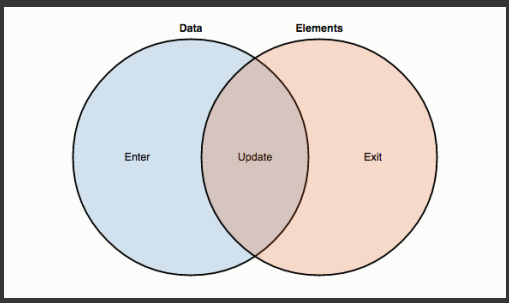
```
var data = [1, 2, 3, 5, 8];
```

```
var data = [
      { x: 10.0, y: 9.14 },
      { x:  8.0, y: 8.14 },
      { x: 13.0, y: 8.74 },
      { x:  9.0, y: 8.77 },
      { x: 11.0, y: 9.26 }
];
```

Data → elements

```
<html>
…
<body>
<svg></svg>
</body></html>

var dataset = [10, 20, 30, 50, 80];

d3.select("svg")          // select where to add new elements
  .selectAll("circle")    // select where to bind data, returns empty selection
    .data(dataset)        // define data mapping
  .enter()                // create new, data-bound placeholder elements
    .append("circle")     // create circle element
    .attr("cx", 50)                                  // static property
    .attr("cy", function(d) { return d; })           // dynamic property
    .attr("r", function(d) { return d / 10 + 1; });  // perform transformations
```

Data → multiple elements



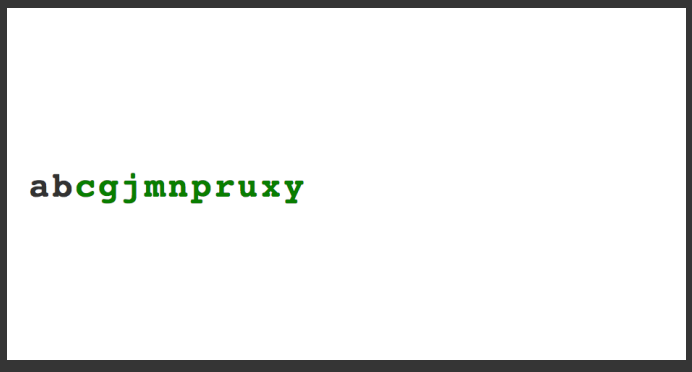
3 sub-selections:

- ***Enter***: New data, missing elements
- ***Update***: Data points joined to existing elements
- ***Exit***: Leftover unbound elements

Enter, update, exit

```
dataset = [10, 70, 120];

var circles = d3.select("svg")
    .selectAll("circle")                    // select where to bind data
    .data(dataset);                         // recompute the join

circles.exit().remove();                    // remove surplus elements

circles.enter().append("circle")            // add new elements
    .attr("cx", 50);

circles                                     // update dynamic properties
    .attr("cy", function(d) { return d; })
    .attr("r", function(d) { return d / 10 + 1; });
```

Update pattern tutorial



https://bl.ocks.org/mbostock/3808218

Setting up

```
<!DOCTYPE html>
<meta charset="utf-8">
<style> /* CSS */</style>
<body>
<script src="https://d3js.org/d3.v3.min.js"></script>

<script>
var alphabet = "abcdefghijklmnopqrstuvwxyz".split("");  // [‘a’, ‘b’, …]

var width = 960,
    height = 500;

var svg = d3.select("body").append("svg")
    .attr("width", width)
    .attr("height", height)
  .append("g")
    .attr("transform", "translate(32," + (height / 2) + ")");
```

Update function

```
var update = function(dataset) {

  // data join
  var text = svg.selectAll("text")
      .data(dataset);

  // update
  text.attr("class", "update");

  // enter
  text.enter().append("text")
      .attr("class", "enter");

  // enter + update
  text
      .text(function(d) { return d; })
      .attr("x", function(d, i) { return i * 32; });

  // exit
  text.exit().remove();
};
```

```
<style>
text {
  font: bold 48px monospace;
}

.enter {
  fill: green;
}

.update {
  fill: #333;
}
</style>
```

Initialize, update on interval

```
// The initial display.
update(alphabet);

// Grab a random sample of letters from the alphabet, in alphabetical order.
setInterval(function() {
  update(d3.shuffle(alphabet)
      .slice(0, Math.floor(Math.random() * 26))
      .sort());
}, 1500);

</script>
```

Key function

Unique identifier of each data entry
Defaults to index

```
var alphabet = [ “a”, “b”, … ]

var update = function(dataset) {

  // data join
  var text = svg.selectAll("text")
      .data(dataset, function(d) { return d; });

  …
}
```

http://bl.ocks.org/mbostock/3808221

Loading Data

d3.csv

stocks.csv

```
symbol,date,price
S&P 500,Jan 2000,1394.46
S&P 500,Feb 2000,1366.42
S&P 500,Mar 2000,1498.58
S&P 500,Apr 2000,1452.43
S&P 500,May 2000,1420.6
S&P 500,Jun 2000,1454.6
S&P 500,Jul 2000,1430.83
```

```
var format = d3.time.format("%b %Y");     //format generator for dates

d3.csv("stocks.csv", function(stocks) {
  stocks.forEach(function(d) {            //array.forEach iterates over rows
    d.price = +d.price;                   //Coerce from strings
    d.date = format.parse(d.date);
  });
});
```

d3.json

stocks.json

```
[{“symbol”: “S&P 500”, "date": "Jan 2000", "price": 1394.46},
  “symbol”: “S&P 500”, "date": ”Feb 2000", "price”: 1366.42},
  “symbol”: “S&P 500”, "date": ”Mar 2000", "price”: 1498.58},
  “symbol”: “S&P 500”, "date": ”Apr 2000", "price”: 1452.43},
  “symbol”: “S&P 500”, "date": ”May 2000", "price”: 1420.6},
  “symbol”: “S&P 500”, "date": ”Jun 2000", "price”: 1454.6},
  “symbol”: “S&P 500”, "date": ”Jul 2000", "price”: 1430.83}...
```

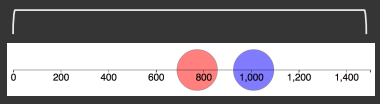
```
var format = d3.time.format("%b %Y");

d3.json("stocks.json", function(stocks) {
  stocks.forEach(function(d) {                    //array.forEach iterates over rows
  d.date = format.parse(d.date);
  });
});
```

Scales

Data space → Visual space

```
var x = d3.scale.linear()
      .domain([0, 1500])
      .range([0, w])

//define your own
function x(d) {
  return d * 0.48;
}
```

```
var data = [{name: "A", price: 1009},
            {name: "B", price: 772}];

var w = 960;
```

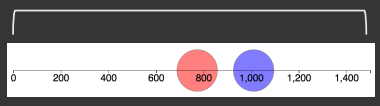



```
var circle = svg.selectAll("circle")
 .data(data)
 .enter()
 .append("circle")
   .attr("cx", function(d) { return x(d);  })
   .attr("cy", 0)
   .attr("r", 50)
   .style("stroke", "black")
   .style("fill", function(d) { return col(d.name);})
   .style("opacity", 0.5);
```

Ordinal mappings

```
var col = d3.scale.ordinal()
    .domain(["A", "B"])
    .range(["blue", "red']);
```

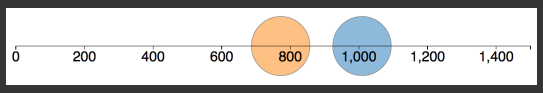
```
var data = [{name: "A", price: 1009},
            {name: "B", price: 772}];

var w = 960;
```




```
var circle = svg.selectAll("circle")
 .data(data)
 .enter()
 .append("circle")
   .attr("cx", function(d) { return x(d);  })
   .attr("cy", 0)
   .attr("r", 50)
   .style("stroke", "black")
   .style("fill", function(d) { return col(d.name);})
   .style("opacity", 0.5);
```

Categorical mappings

```
var col = d3.scale.category10()
   .domain(["A", "B"]);
```



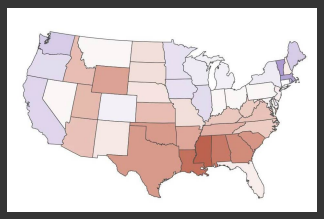

```
var col = d3.scale.ordinal()
   .range(colorbrewer.Set1[9]]);
```

Interpolators (quantitative scales)

```
var x = d3.scale.linear()
   .domain([12, 24])
   .range(["steelblue", "brown"]);

x(16); //#666586
```

```
var x = d3.scale.linear()
   .domain([12, 24])
   .range(["0px", "720px"]);

x(16); //240px
```

Diverging scale



```
var col = d3.scale.linear()
   .domain([0, 50, 100])
   .range(["blue", "white", "red"]);
```

Axes

Creating and rendering an axis

```
var x = d3.scale.linear()
      .domain([0, 1500])
      .range([0, w])
```

Define axis element

```
var xAxis = d3.svg.axis()
     .scale(x);
```

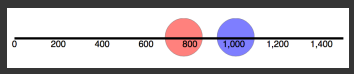
Render by calling a <g> selection

```
svg.append("g")
  .attr("class", "x axis")
  .call(xAxis);
```

Creating and rendering an axis

Customize using CSS

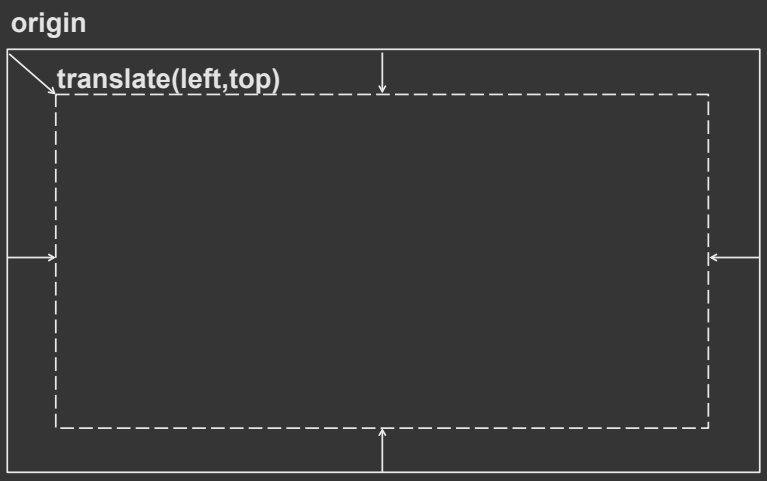
```
.axis path, .axis line {
  fill: none;
  stroke: #000;
  shape-rendering: crispEdges;
}
```




```
var xAxis = d3.svg.axis()
    .scale(y)
    .ticks(4);
```

SVG Coordinate System

SVG Coordinates



Use transforms on <g> to define a new origin (e.g., plotting area)

Axis example

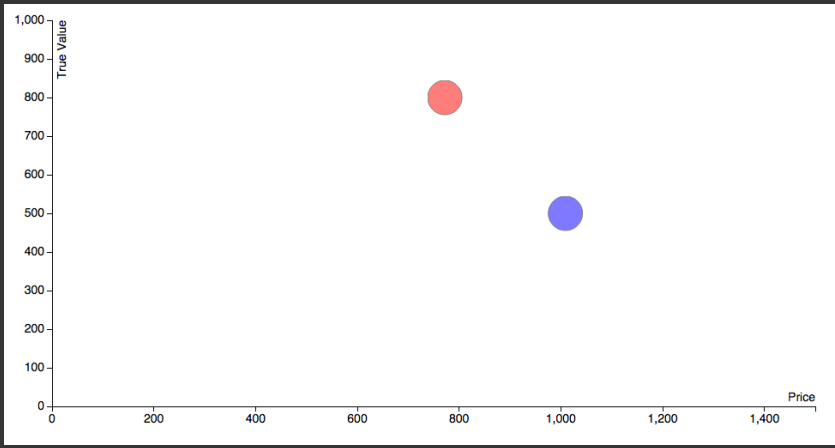
```
var data = [{name: "A", price: 1009},{Value: 500},
            {name: "B", price: 772}],Value: 900}];
```

```
var w = 960;
```


```
var x = d3.scale.linear()
      .domain([0, 2000])
      .range([0, w])

var circle = svg.selectAll("circle")
 .data(data)
 .enter()
 .append("circle")
   .attr("cx", function(d) { return x(d);  })
   .attr("cy", 0)
   .attr("r", 50)
   .style("stroke", "black")
   .style("fill", function(d) { return col(d.name);})
   .style("opacity", 0.5);
```

```
var xAxis = d3.svg.axis()
      .scale(x);

svg.append("g")
   .attr("class", "x axis")
   .call(xAxis);
```

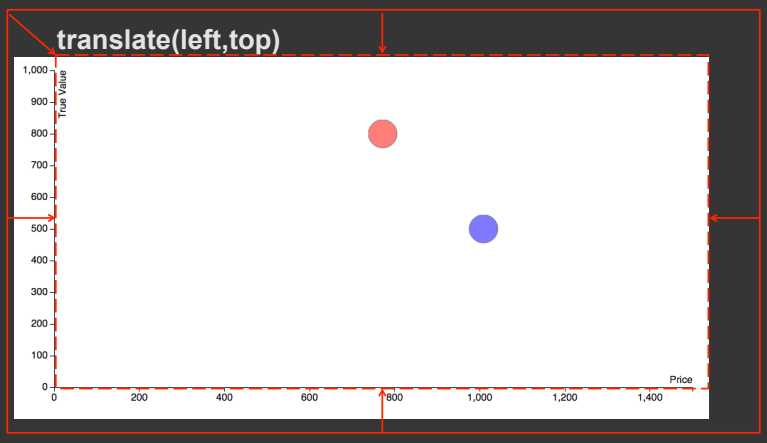
Let's make this!



Transform on <g> attribute

origin

translate(left,top)



Define a new origin for plotting area
Axes appear in margin

Transform on <g> attribute

```
<!DOCTYPE html>
<meta charset="utf-8">
<style> /* CSS */</style>
<body>

<script src="http://d3js.org/d3.v3.min.js"></script>
<script>

var margin = {top: 20, right: 20, bottom: 30, left: 50},
        w = 960 – margin.left – margin.right,
        h = 500 – margin.top – margin.bottom;


var svg = d3.select("body").append("svg")
   .attr("width", w + margin.left + margin.right)
   .attr("height", h + margin.top + margin.bottom)
  .append("g")
   .attr("transform", "translate(" + margin.left + "," + margin.top + ")");
```

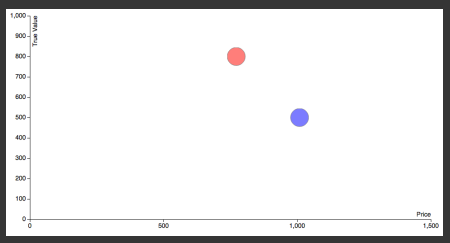
Create the axes, marks

```
var x = d3.scale.linear()
      .domain([0, 1500])
      .range([0, w])

var xAxis = d3.svg.axis()
      .scale(x)
      .orient("bottom");


var y = d3.scale.linear()
      .domain([0, 1000])
      .range([h, 0])

var yAxis = d3.svg.axis()
      .scale(y)
      .orient("left");
```

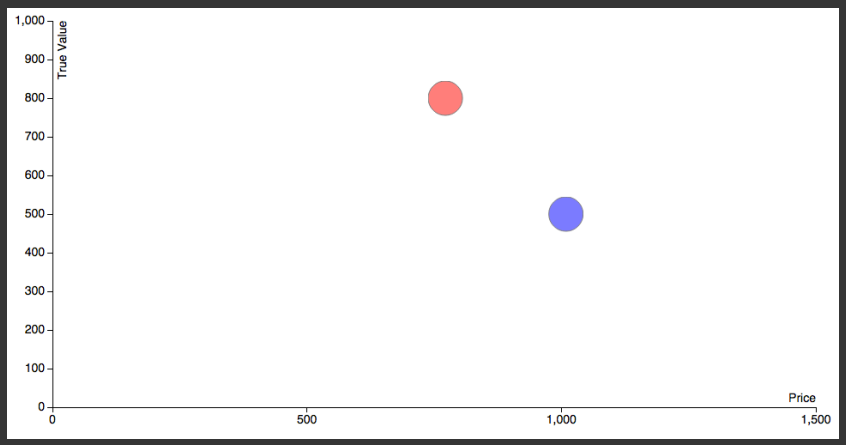
```
var col = d3.scale.ordinal()
   .domain(["A", "B"])
   .range(['"blue", "red']);


 var data = [{name: "A", price: 1009, tValue: 500},
             {name: "B", price: 772, tValue: 900}];


var circle = svg.selectAll("circle")
 .data(data)
 .enter()
 .append("circle")
   .attr("cx", function(d) { return x(d.price);  })
   .attr("cy", function(d) { return y(d.tValue); })
   .attr("r", 50)
   .style("stroke", "black")
   .style("fill", function(d) { return col(d.name);})
   .style("opacity", 0.5);
```

Add the axes

```
svg.append(“g”)
        .attr(“class”, “x axis”)
        .attr("transform", "translate(0," + h + ")”)
        .call(xAxis);
    .append("text”)
        .attr("class", "label")
        .attr("x", w)
        .attr("y", -6)
        .style("text-anchor", "end")
        .text("Price");

svg.append("g")
        .attr("class", "y axis")
        .call(yAxis)
    .append("text")
        .attr("class", "label")
        .attr("transform", "rotate(-90)")
        .attr("y", 6)
        .style("text-anchor", "end")
        .text("True Value");
</script>
```



Voila!

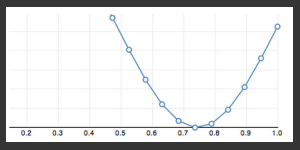


Path Generators

```
<path d="M152.64962091501462,320.5600780855698L133.
88913955606318,325.4363177123538L134.96890954443046
,330.37917634921996L131.19348249532786,331.15839361
4812L98.56681109628815,335.53933807857004L91.144507
99488135,333.79662025279L72.1880101321918,333.74733
970068166L69.51723455785742,332.8569681440152L62.37
313911354066,333.2100666843387L62.248334309137434,3
35.3677272708405L58.843440998888326,335.05749596050
36L53.97667317214221,331.36075125633175L56.30952738
```

d3.svg.line

Path defined by *x* and *y*



```
var x = d3.scale.linear(),
    y = d3.scale.linear();

var line = d3.svg.line()
   .x(function(d) { return x(d.x); })
   .y(function(d) { return y(d.y); });
```

```
var objects = [(0.47,0.55), (0.52,0.4), …]

svg.append("path")
   .datum(objects)
   .attr("class", "line")
   .attr("d", line);
```

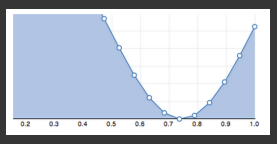
Append closepath (Z) to close

```
g.append("path")
   .attr("d", function(d) { return line(d) + "Z"; });
```

Linear, step, and basis interpolation

d3.svg.area

Path defined by *x*, y_0, and y_1



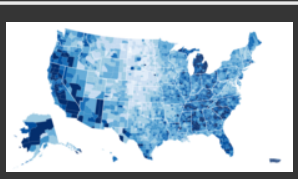
```
var x = d3.scale.linear(),
    y = d3.scale.linear();

var area = d3.svg.area ()
   .x(function(d) { return x(d.x); })
   .y0(height)
   .y1(function(d) { return y(d.y); });
```

For non-stacked area charts, y_0 is constant

d3.geo.path

Like d3 line

GeoJSON/TopoJSON format



```
var projection = d3.geo.albersUsa()
    .scale(1280)
    .translate([width / 2, height / 2]);

var path = d3.geo.path()
    .projection(projection);
```

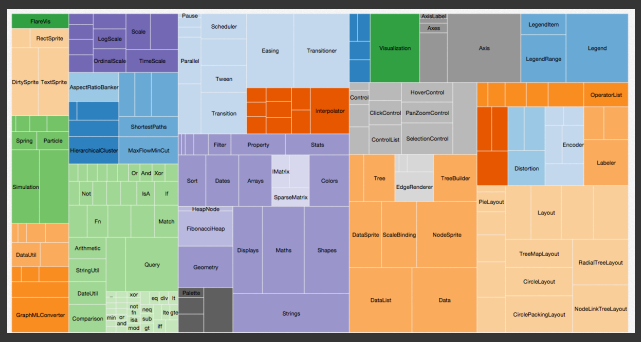
Other path generators

- *d3.svg.line* - create a new line generator
- *d3.svg.line.radial* - create a new radial line generator
- *d3.svg.area* - create a new area generator
- *d3.svg.area.radial* - create a new radial area generator
- *d3.svg.arc* - create a new arc generator
- *d3.svg.symbol* - create a new symbol generator
- *d3.svg.chord* - create a new chord generator
- *d3.svg.diagonal* - create a new diagonal generator
- *d3.svg.diagonal.radial* - create a new radial diagonal generator

Layouts

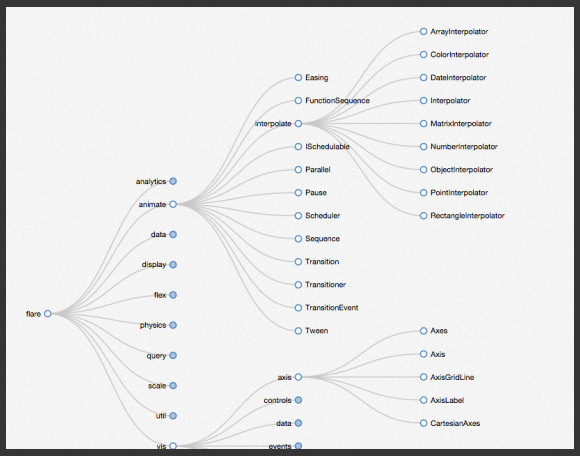
Hierarchical layouts

d3.layout.treemap



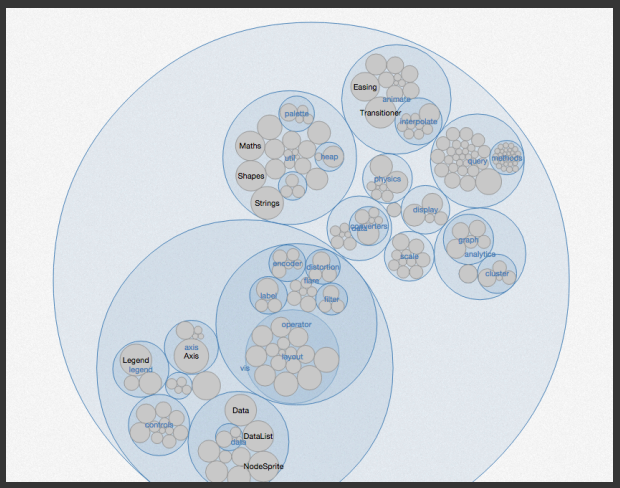
Hierarchical layouts

d3.layout.tree



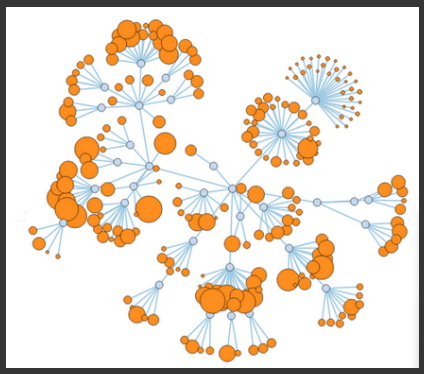
Hierarchical layouts

d3.layout.pack



Network layout

d3.layout.force



Resources

D3 API Documentation at https://github.com/mbostock/d3/wiki,
D3 wiki, D3 google group

Example code
- Mike Bostock
- Also Scott Murray, Jerome Cukier

YouTube tutorials
- D3.js tutorial series

Interaction Resources for D3

Write functions to update the visualization on mouse events

```
var circle = svg.selectAll("circle")
 .data(data)
 .enter()
 .append("circle")
   .attr("cx", function(d) { return x(d.price);  })
   .attr("cy", function(d) { return y(d.tValue); })
   .attr("r", 50)
   .style("stroke", "black")
   .style("fill", function(d) { return col(d.name);})
   .style("opacity", 0.5)
   .on("mouseover", function(d,i){ showDetails(i); })
   .on("mouseout", function(d,i){ hideDetails(i); });
```

CSS can simplify simple interactions

```
.circle:hover {
  fill: yellow;
}
```

Interaction Resources for D3

Use HTML inputs or JavaScript widgets as needed

- e.g., http://www.d3noob.org/2014/04/using-html-inputs-with-d3js.html

See d3.behaviors for drag and zoom

- **Zoom example:** http://bl.ocks.org/mbostock/9656675
- **Drag + zoom:** http://bl.ocks.org/mbostock/6123708

Use transition() for smooth animations between states

- http://blog.visual.ly/creating-animations-and-transitions-with-d3-js/

```
circle.transition()
  .attr("r",40)
  .duration(1000)
  .delay(100)
```