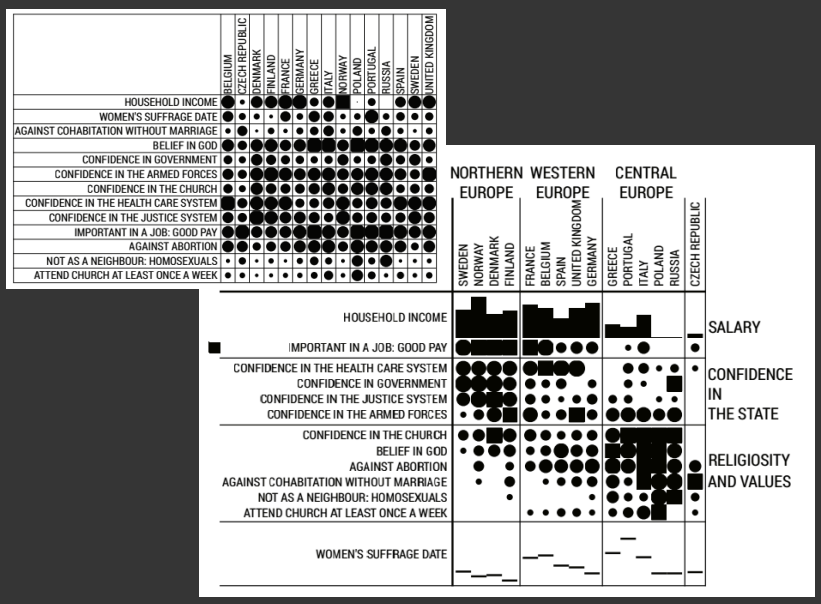
D3 Introduction

Maneesh Agrawala
Jessica Hullman

CS 294-10: Visualization
Fall 2014

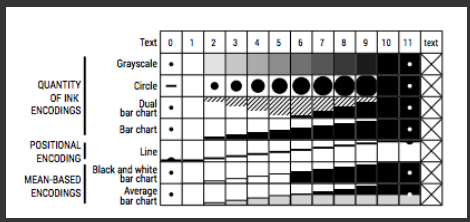
From Interaction II (last week)



[Perin, Dragicevic, and Fekete 2014]

Visual encodings

Visual encodings ensure that visual differences are roughly proportional to numerical differences.

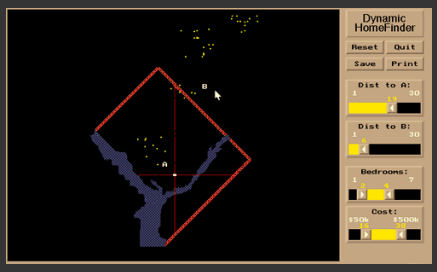


Announcements

Assignment 3: Visualization Software

Create a **small** interactive visualization application – you choose data domain and visualization technique.

1. Describe data and storyboard interface
2. Implement interface and produce final writeup
3. Submit the application and a final writeup on the wiki



Can work alone or in pairs

Final write up due before class on **Oct 15, 2014**

D3 Introduction

Topics

Motivation
Getting started

Selections

Scales
Axes
Coordinate system
Path generators
Layouts

[Adapted from Mike Bostock's D3 Workshop]

Motivation

Visualization with Web Standards

Transformation, not representation (HTML, SVG)

Constructing a DOM from data

Benefits:
Expressivity
Debugging tools
Better documentation

hello-world.html

```
<!DOCTYPE html>
<meta charset="utf-8">
<body>
Hello, world!
```

hello-svg.html

```
<!DOCTYPE html>
<meta charset="utf-8">
<svg width="960" height="500">
  <text x=”10” y=”10”>
    Hello, world!
  </text>
</svg>!
```

hello-css.html

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

hello-javascript.html

```
<!DOCTYPE html>
<meta charset="utf-8">
<script>
console.log("Hello, world!");
</script>
```

hello-d3.html

```
<!DOCTYPE html>
<meta charset="utf-8">
<style> /* CSS */</style>
<body>
<script src="d3.v2.js"></script>
```

Getting Started

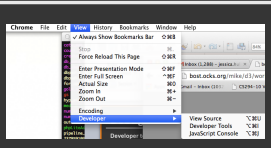
Running locally

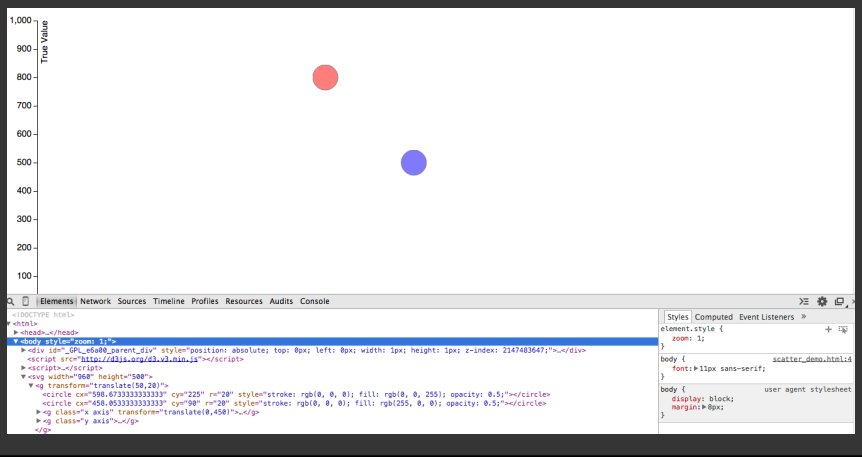
> python -m SimpleHTTPServer 8888 &

http://localhost

Debugging tools

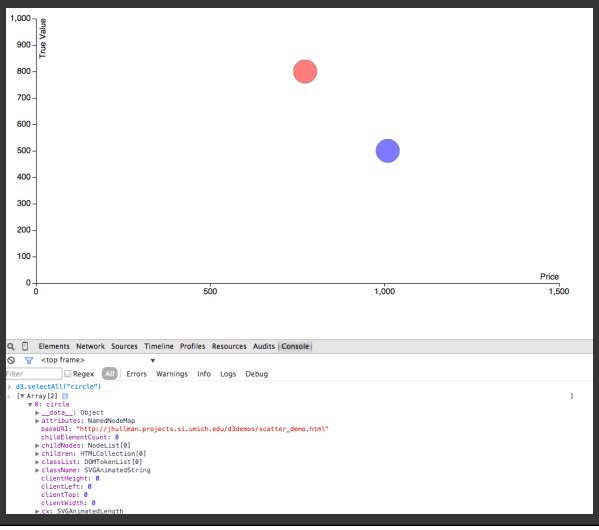
Developer tools





Debugging tools

JavaScript console



Selections

Operating on a selection

```
var ps = document.getElementsByTagName("p");
for (var i = 0; i < ps.length; i++) {
  var p = ps.item(i);
  p.style.setProperty("color", "white", null);
}
```
a

```
p { color: white; }
```
b

```
$("p").css("color", "white");
```
c

```
d3.selectAll("p").style("color", "white");
```
d

Selections in d3 are associated with operators to set properties.

Select SVG circles

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


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


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

Other basic shapes

```
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);
```

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 body = d3.selectAll("body");

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

Selects multiple elements, adds one element to each.

Data → multiple elements

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

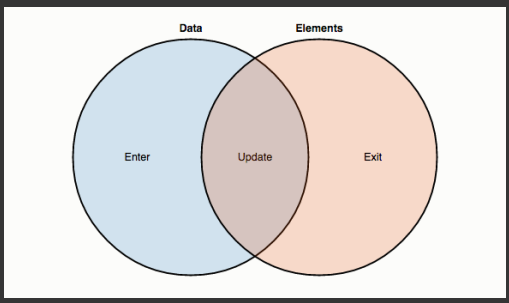
```
var data = [                            //object
      {x: 10.0, y: 9.14},
      {x:  8.0, y: 8.14},
      {x: 13.0, y: 8.74},
      {x:  9.0, y: 8.77},
      {x: 11.0, y: 9.26}
      ];
```

Data → multiple elements

```
svg.selectAll("circle")
    .data(data)                          //data join
  .enter().append("circle")
    .attr("cx", x)
    .attr("cy", y)
    .attr("r", 2.5);
```

D3's data join: Defines enter, update, and exit subselections

Data → multiple elements



3 selections:

- *Enter*: Missing elements
- *Update*: Data points joined to existing elements
- *Exit*: Leftover unbound elements

Data → multiple elements

```
var circle = svg.selectAll("circle")
    .data(data)

circle.enter().append("circle")
      .attr("cx", x)
      .attr("cy", y)
      .attr("r", 2.5);
```

//Returns a new empty selection

//Join the selection to data: 3 new selections (enter, update, exit)

//Appending to the enter selection adds the missing elements to the SVG container

Accessor functions: `function x(d) { return d.x; }`

Why joins?

Enter, update, exit

```
var circle = svg.selectAll("circle")
    .data(data)

circle.exit().remove()

circle.enter().append("circle")
      .attr("r", 2.5);

circle
      .attr("cx", x)
      .attr("cy", y)
```

//Selecting circles

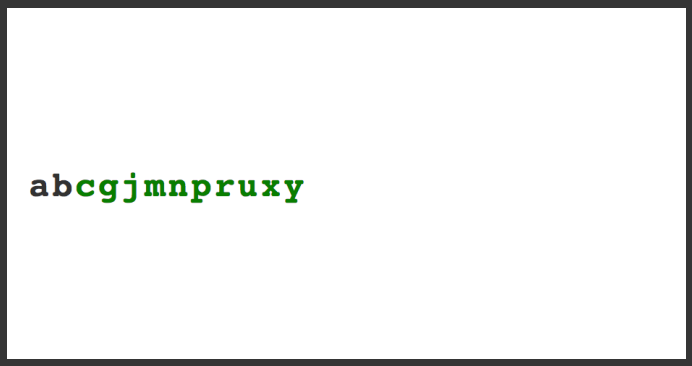
//Recompute the join

//Remove surplus elements

//Add new elements (set constant attribute)

//Update the x and y position with the new data

Update pattern tutorial



Setting things up

```
<!DOCTYPE html>
<meta charset="utf-8">
<style> /* CSS */</style>
<body>
<script src="d3.v2.js"></script>

<script>
var alphabet = "abcdefghijklmnopqrstuvwxyz".split("");

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) + ")");
```

Initialize, update on interval

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

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

// Shuffles the input array.
function shuffle(array) {
  var m = array.length, t, i;
  while (m) {
    i = Math.floor(Math.random() * m--);
    t = array[m], array[m] = array[i], array[i] = t;
  }
  return array;
}

</script>
```

Update function

```
function update(data) {

  // DATA JOIN
  var text = svg.selectAll("text")
      .data(data);

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

  // ENTER
  text.enter().append("text")
      .attr("class", "enter")
      .attr("x", function(d, i) { return i * 32; })
      .attr("dy", ".35em");

  // ENTER + UPDATE
  text.text(function(d) { return d; });

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

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

        .enter {
                fill: green;
        }

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

Entering letters at end

```
function update(data) {

  // DATA JOIN
  var text = svg.selectAll("text")
      .data(data);

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

  // ENTER
  text.enter().append("text")
      .attr("class", "enter")
      .attr("x", function(d, i) { return i * 32; })
      .attr("dy", ".35em");

  // ENTER + UPDATE
  text.text(function(d) { return d; });

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

Initialize, update on interval

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

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

// Shuffles the input array.
function shuffle(array) {
  var m = array.length, t, i;
  while (m) {
    i = Math.floor(Math.random() * m--);
    t = array[m], array[m] = array[i], array[i] = t;
  }
  return array;
}

</script>
```

Key function

Defaults to index

To update same letter each time:

```
var alphabet = [
  {name: “a”, val: “a”},
  {name: “b”, val: “b”},
  {name: “c”, val: “c”}, …

function key(d) { return d.name; }

function update(data) {

  // DATA JOIN
  var text = svg.selectAll("text")
      .data(data, key);
```

Scatterplot example

```
var data = [
  {name: "Alice", x: 10.0, y: 9.14},
  {name:   "Bob", x:  8.0, y: 8.14},
  {name: "Carol", x: 13.0, y: 8.74},
  {name:  "Dave", x:  9.0, y: 8.77},
  {name: "Edith", x: 11.0, y: 9.26}
];

function key(d) { return d.name; }
function x(d) { return d.x; }
function y(d) { return d.y; }

var circle = svg.selectAll("circle")
    .data(data, key)
    .attr("cx", x)
    .attr("cy", y)
    .attr("r", 2.5);
```

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}...
```

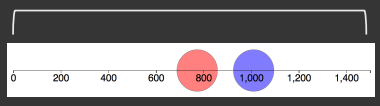
```
var format = d3.time.format("%b %Y");

d3.csv("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;
```



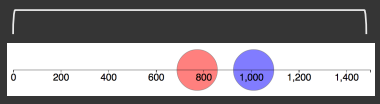

```
var circle = svg.selectAll("circle")
 .data(data)
 .enter()
 .append("circle")
   .attr("cx", function(d) { return x(d);  })
   .attr("cy", 0)
   .attr("r", 50)
   .style("stroke", "black")
   .style("fill", function(d) { return col(d.name);})
   .style("opacity", 0.5);
```

Ordinal mappings

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

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

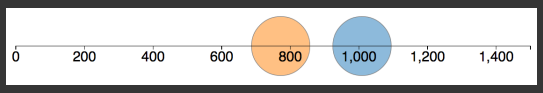
```
var w = 960;
```



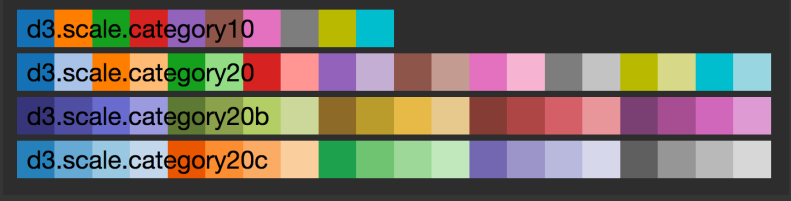

```
var circle = svg.selectAll("circle")
 .data(data)
 .enter()
 .append("circle")
   .attr("cx", function(d) { return x(d);  })
   .attr("cy", 0)
   .attr("r", 50)
   .style("stroke", "black")
   .style("fill", function(d) { return col(d.name);})
   .style("opacity", 0.5);
```

Categorical mappings

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




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

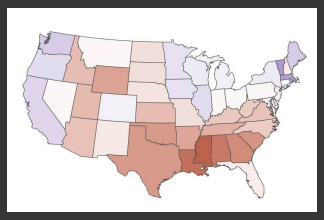


Interpolators (quantitative scales)

```
var col = 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);
```

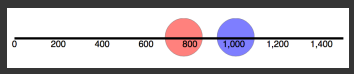
Render by calling a <g> selection

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

Creating and rendering an axis

Customize using CSS

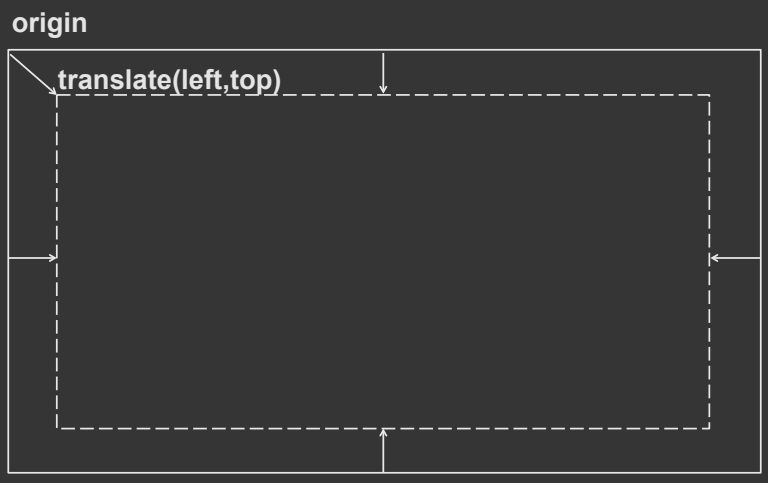
```
.axis path, .axis line {
  fill: none;
  stroke: #000;
  shape-rendering: crispEdges;
}
```




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

SVG Coordinate System

SVG Coordinates

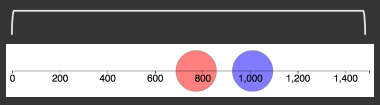


Use transforms on <g> to define a new origin (e.g., plotting area)

Axis example

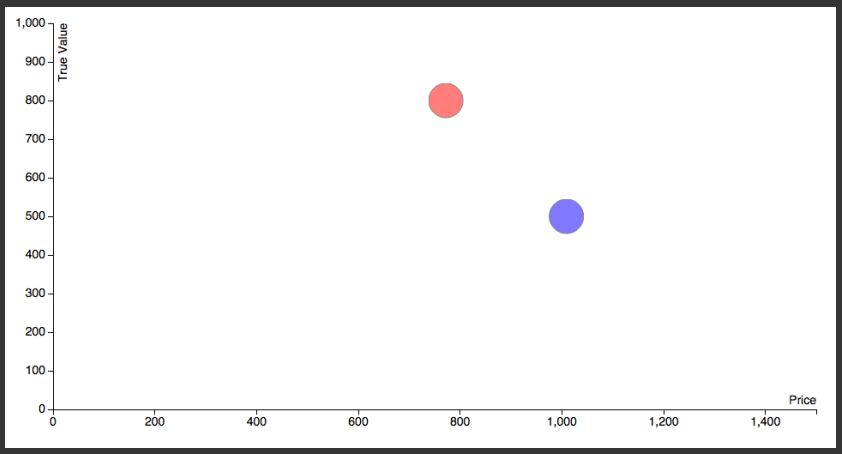
```
var data = [{name: "A", price: 1009},{Value: 500},
            {name: "B", price: 772},]Value: 900}];
```

```
var w = 960;
```



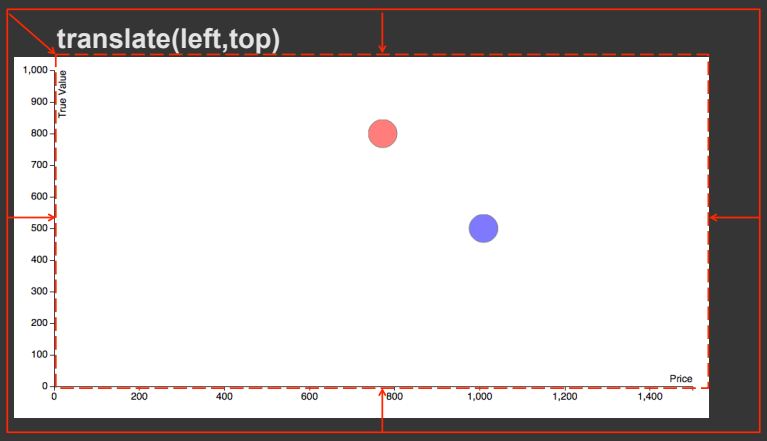

```
var x = d3.scale.linear()
      .domain([0, 2000])
      .range([0, w])

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);
```



Transform on <g> attribute

origin



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 + ")");
```

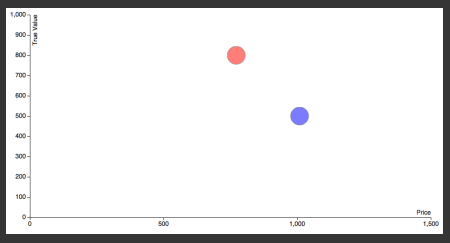
Create the axes, marks

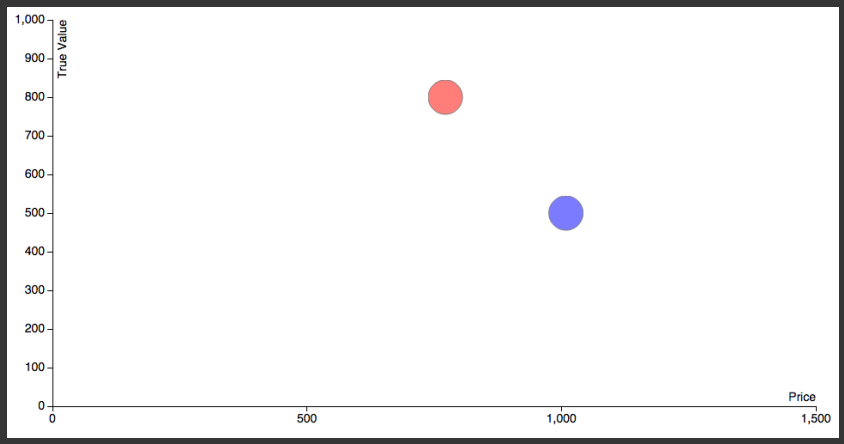
```
var x = d3.scale.linear()
        .domain([0, 1500])
        .range([0, w])

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>
```



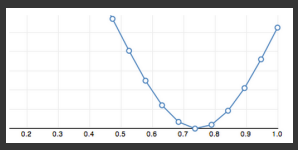


Path Generators

```
<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
36L53.97667317214221,331.360751256331 75L56.30952738
```

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);
```

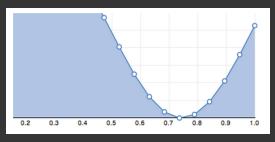
Append closepath (Z) to close

```
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



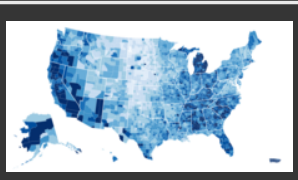
```
var x = d3.scale.linear(),
    y = d3.scale.linear();

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);
```

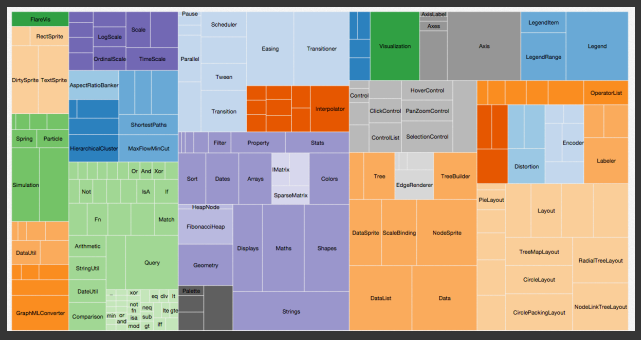
Other path generators

- *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

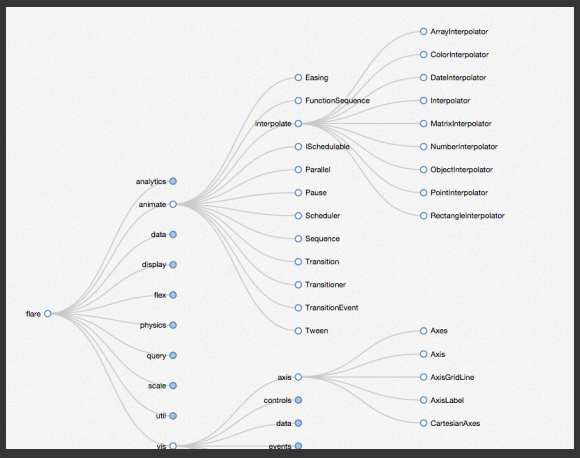
Hierarchical layouts

```
var treemap = d3.layout.treemap()
    .padding(4)
    .size([width, height]);

var parent = {"children": [...]},
    child = {"value": ...};
```



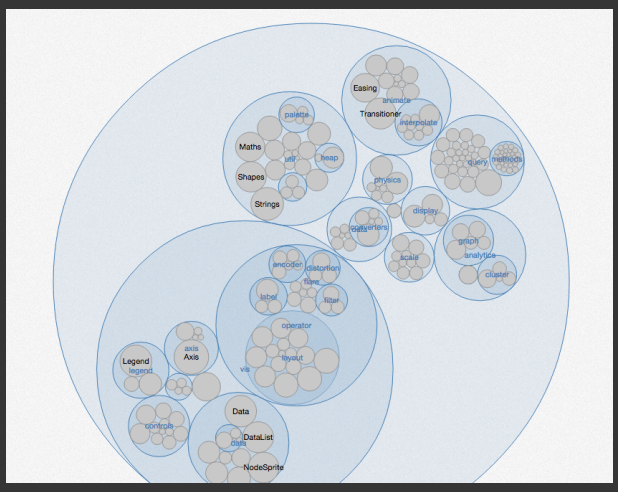
Hierarchical layouts

d3.layout.tree



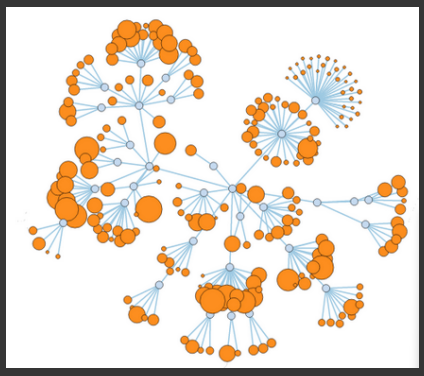
Hierarchical layouts

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)
```