Lecture 9

D3.js

Recap

- HTML Canvas
  - Comparison to SVG browser graphics
  - Drawing shapes
  - Basic Trigonometry
  - Basic collision detection
  - Animation and User Interaction
    - Examples
  - Advanced collision handling
  - Linear Algebra and Transformations
- SVG

Recap

- Dealing with matrices

Today

- D3.js
  - http://thenextweb.com/dd/2015/04/21/the-14-best-data-visualization-tools/
  - http://www.ted.com/talks/hans_rosling_shows_the_best_stats_you_ve_ever_seen
  - http://www.nytimes.com/interactive/2012/02/13/us/politics/2013-budget-proposal-graphic.html
  - http://www.nytimes.com/interactive/2013/05/25/sunday-review/corporate-taxes.html

D3.js - Overview

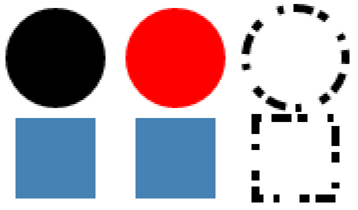
- Purpose:
  - D3 (standing for Data-Driven Documents) is a JavaScript library for producing dynamic, interactive data visualizations in web browsers.
- Capabilities:
  - Principal functions allow for selections, transitions and data-binding.
    - D3 can select elements in the DOM programmatically (can use instead of the verbose DOM API or jQuery)
    - By declaring a transition, styles can be smoothly interpolated/animated over a certain time.
    - Large datasets (e.g. JSON/XML format) can be easily bound to SVG objects using simple D3.js functions to generate rich text/graphic charts and diagrams.
- Technologies:
  - Built on top of common web standards like HTML, CSS, the DOM and SVG (can also use with HTML5 Canvas).
    - Exportable nature of SVG enables graphics created by D3 to be used in print publications.
- Origins:
  - Initially released in 2011 (successor to the earlier Protovis framework) – Now used widely (e.g. by the New York Times, OpenStreetMap etc.)
    - Recent version - v4 has big breaks in the API from previous versions

D3.js – SVG recap

- SVG (Scalable Vector Graphics) is an XML format used for drawing.
- SVG also has a DOM – there are elements with parents and children and attributes, and you can respond to the same mouse/touch events.
- CSS styles and selectors can apply to SVG elements. E.g.

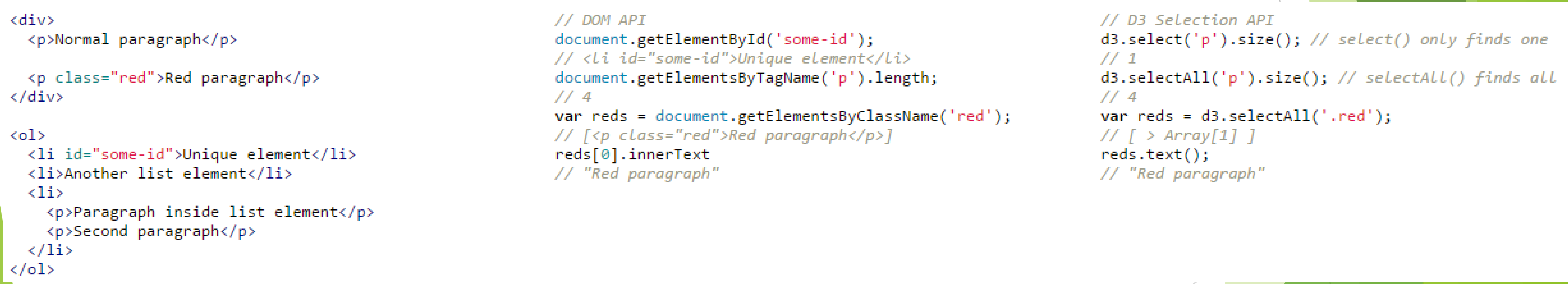
```
<svg width="300" height="180">
  <circle cx="30"  cy="50" r="25" />
  <circle cx="90"  cy="50" r="25" class="red" />
  <circle cx="150" cy="50" r="25" class="fancy" />

  <rect x="10"  y="80" width="40" height="40"
    fill="steelBlue" />
  <rect x="70"  y="80" width="40" height="40"
    style="fill: steelBlue" />
  <rect x="130" y="80" width="40" height="40"
    class="fancy" />
</svg>
```

```
.red {
  fill: red; /* not background-color! */
}

.fancy {
  fill: none;
  stroke: black; /* similar to border-color */
  stroke-width: 3pt; /* similar to border-width */
  stroke-dasharray: 3,5,10;
}
```



D3.js - Selectors

- SVG has a DOM (since XML based)
  - Can attach CSS styles and selectors to SVG elements

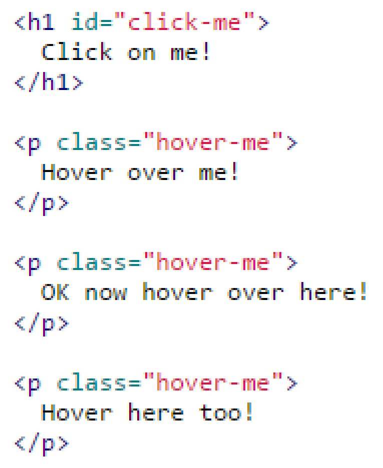


```html
<div>
  <p>Normal paragraph</p>

  <p class="red">Red paragraph</p>
</div>

<ol>
  <li id="some-id">Unique element</li>
  <li>Another list element</li>
  <li>
    <p>Paragraph inside list element</p>
    <p>Second paragraph</p>
  </li>
</ol>
```

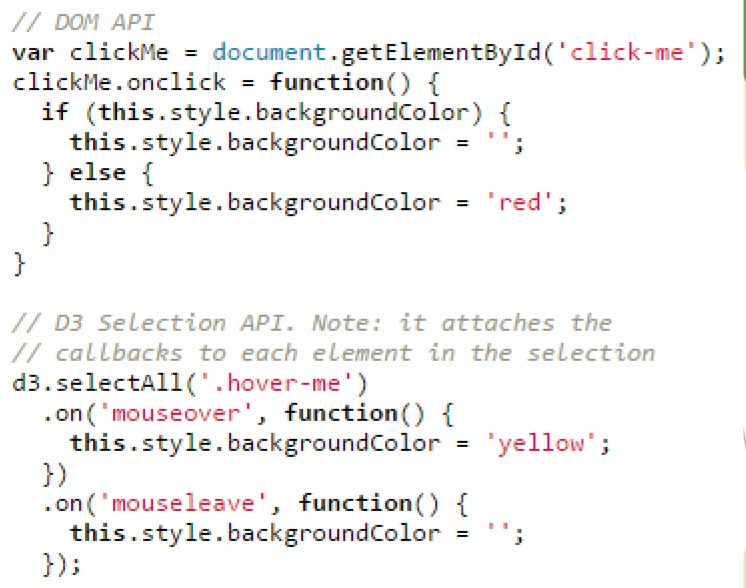
```javascript
// DOM API
document.getElementById('some-id');
// <li id="some-id">Unique element</li>
document.getElementsByTagName('p').length;
// 4
var reds = document.getElementsByClassName('red');
// [<p class="red">Red paragraph</p>]
reds[0].innerText
// "Red paragraph"
```

```javascript
// D3 Selection API
d3.select('p').size(); // select() only finds one
// 1
d3.selectAll('p').size(); // selectAll() finds all
// 4
var reds = d3.selectAll('.red');
// [ > Array[1] ]
reds.text();
// "Red paragraph"
```

D3.js – Event Listeners



```html
<h1 id="click-me">
  Click on me!
</h1>

<p class="hover-me">
  Hover over me!
</p>

<p class="hover-me">
  OK now hover over here!
</p>

<p class="hover-me">
  Hover here too!
</p>
```

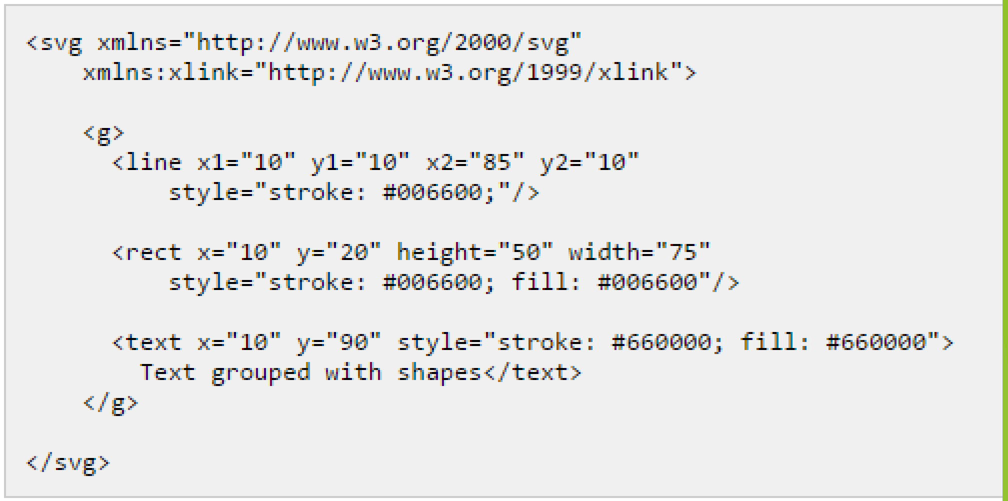


```javascript
// DOM API
var clickMe = document.getElementById('click-me');
clickMe.onclick = function() {
  if (this.style.backgroundColor) {
    this.style.backgroundColor = '';
  } else {
    this.style.backgroundColor = 'red';
  }
}

// D3 Selection API. Note: it attaches the
// callbacks to each element in the selection
d3.selectAll('.hover-me')
  .on('mouseover', function() {
    this.style.backgroundColor = 'yellow';
  })
  .on('mouseleave', function() {
    this.style.backgroundColor = '';
  });
```

- Like jQuery, can attach listeners (click, mouseover etc.)
  - Above are some examples of adding listeners to the click, mouseover and mouseleave events.
- In D3 (like with jQuery), the methods on the selection can chain (that is, they return themselves, so we can group them visually).

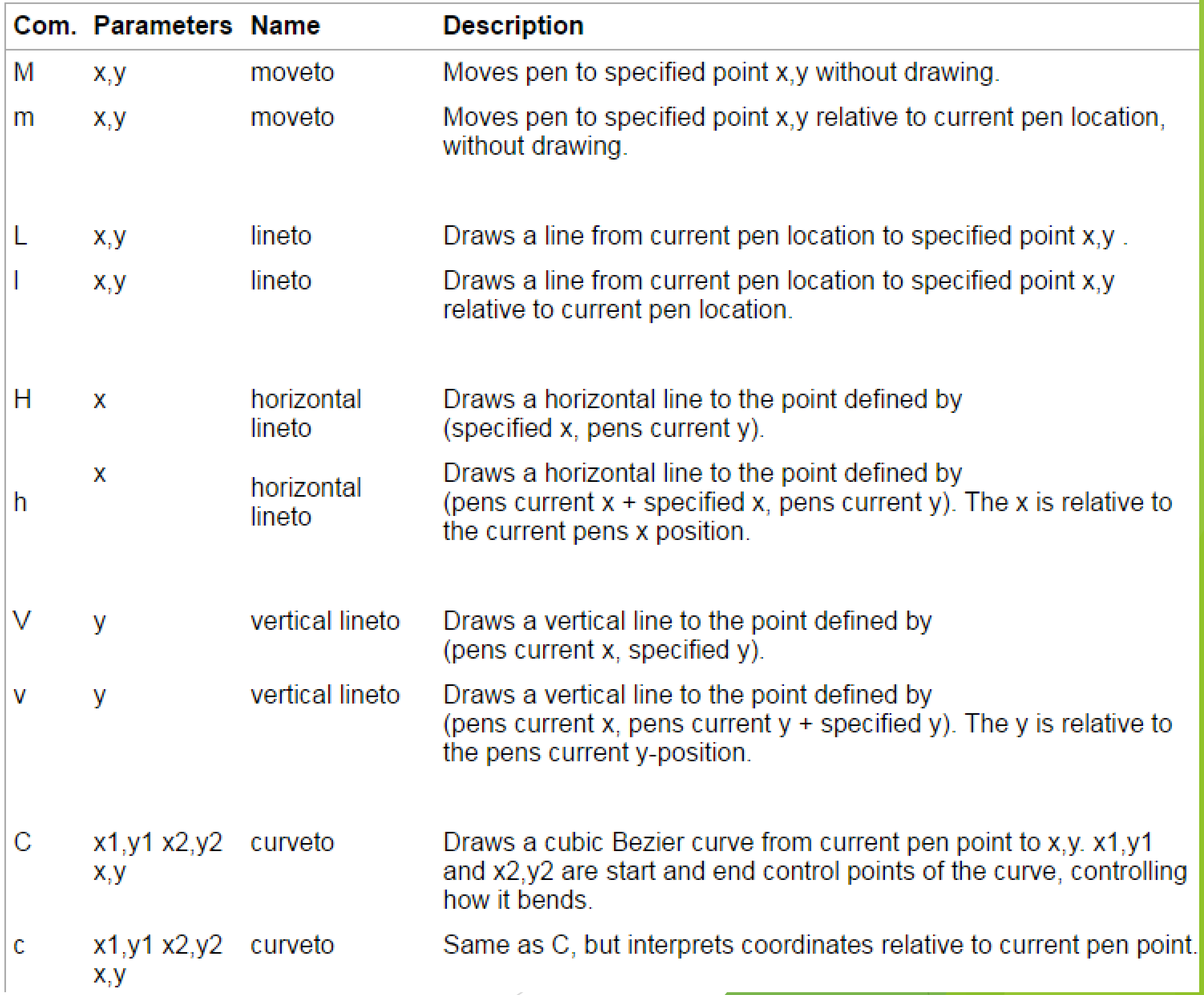
D3.js – SVG Tags

- Where HTML has the <div> and <span> tags, SVG has the <g> tag for an arbitrary group.
  - <g> is used a lot in D3 - Great for applying styles to a group (including re-positioning the groups).
- The <text> tag is good for simple labels.
- The <path> tag s the most powerful element in the SVG library of basic shapes. It is powerful but complex, it can be used for either lines or arbitrary filled-in shapes depending on the styling.



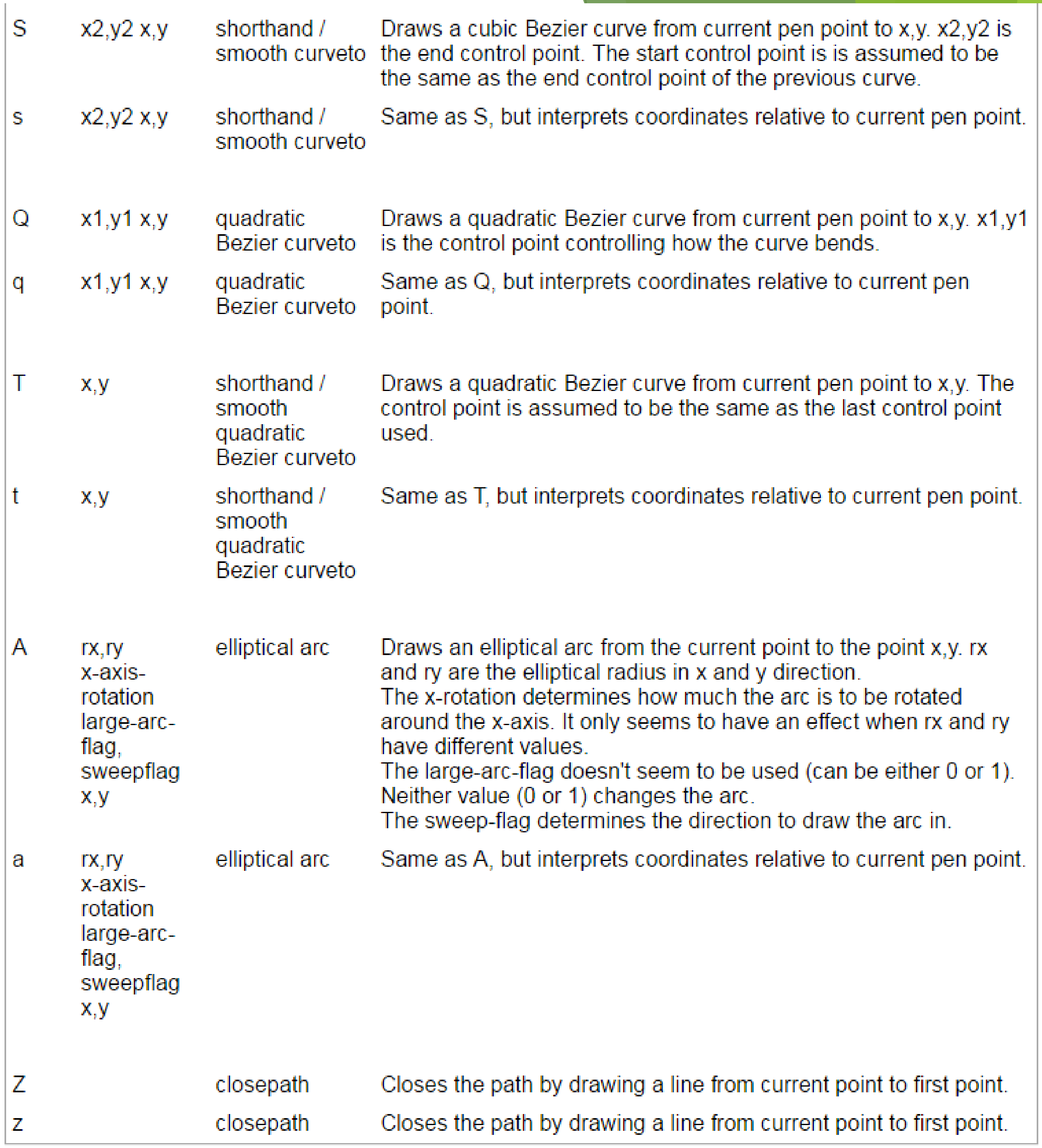
```
<svg xmlns="http://www.w3.org/2000/svg"
    xmlns:xlink="http://www.w3.org/1999/xlink">

    <g>
      <line x1="10" y1="10" x2="85" y2="10"
          style="stroke: #006600;"/>

      <rect x="10" y="20" height="50" width="75"
          style="stroke: #006600; fill: #006600"/>

      <text x="10" y="90" style="stroke: #660000; fill: #660000">
        Text grouped with shapes</text>
    </g>

</svg>
```

SVG Path Commands



Com.	Parameters	Name	Description
M	x,y	moveto	Moves pen to specified point x,y without drawing.
m	x,y	moveto	Moves pen to specified point x,y relative to current pen location, without drawing.
L	x,y	lineto	Draws a line from current pen location to specified point x,y .
l	x,y	lineto	Draws a line from current pen location to specified point x,y relative to current pen location.
H	x	horizontal lineto	Draws a horizontal line to the point defined by (specified x, pens current y).
h	x	horizontal lineto	Draws a horizontal line to the point defined by (pens current x + specified x, pens current y). The x is relative to the current pens x position.
V	y	vertical lineto	Draws a vertical line to the point defined by (pens current x, specified y).
v	y	vertical lineto	Draws a vertical line to the point defined by (pens current x, pens current y + specified y). The y is relative to the pens current y-position.
C	x1,y1 x2,y2 x,y	curveto	Draws a cubic Bezier curve from current pen point to x,y. x1,y1 and x2,y2 are start and end control points of the curve, controlling how it bends.
c	x1,y1 x2,y2 x,y	curveto	Same as C, but interprets coordinates relative to current pen point.

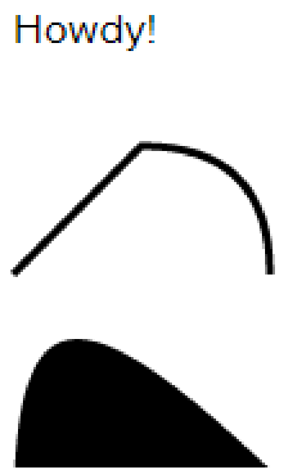
SVG Path Commands



S	x2,y2 x,y	shorthand / smooth curveto	Draws a cubic Bezier curve from current pen point to x,y. x2,y2 is the end control point. The start control point is is assumed to be the same as the end control point of the previous curve.
s	x2,y2 x,y	shorthand / smooth curveto	Same as S, but interprets coordinates relative to current pen point.
Q	x1,y1 x,y	quadratic Bezier curveto	Draws a quadratic Bezier curve from current pen point to x,y. x1,y1 is the control point controlling how the curve bends.
q	x1,y1 x,y	quadratic Bezier curveto	Same as Q, but interprets coordinates relative to current pen point.
T	x,y	shorthand / smooth quadratic Bezier curveto	Draws a quadratic Bezier curve from current pen point to x,y. The control point is assumed to be the same as the last control point used.
t	x,y	shorthand / smooth quadratic Bezier curveto	Same as T, but interprets coordinates relative to current pen point.
A	rx,ry x-axis-rotation large-arc-flag, sweepflag x,y	elliptical arc	Draws an elliptical arc from the current point to the point x,y. rx and ry are the elliptical radius in x and y direction. The x-rotation determines how much the arc is to be rotated around the x-axis. It only seems to have an effect when rx and ry have different values. The large-arc-flag doesn't seem to be used (can be either 0 or 1). Neither value (0 or 1) changes the arc. The sweep-flag determines the direction to draw the arc in.
a	rx,ry x-axis-rotation large-arc-flag, sweepflag x,y	elliptical arc	Same as A, but interprets coordinates relative to current pen point.
Z		closepath	Closes the path by drawing a line from current point to first point.
z		closepath	Closes the path by drawing a line from current point to first point.

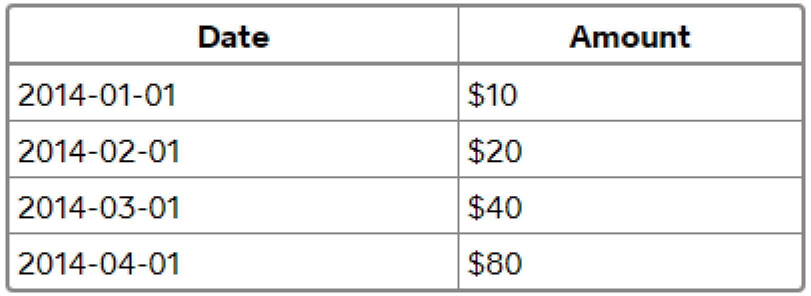
D3.js – SVG

```
<svg width="300" height="180">
  <g transform="translate(5, 15)">
    <text x="0" y="0">Howdy!</text>
  </g>

  <g transform="translate(5, 55)">
    <!-- M: move to (jump)
         L: line to
         Q: curve to (quadratic) -->
    <path d="M0,50 L50,0 Q100,0 100,50"
      fill="none" stroke-width="3" stroke="black" />
  </g>

  <g transform="translate(5, 105)">
    <!-- C: curve to (cubic)
         Z: close shape -->
    <path d="M0,100 C0,0 25,0 125,100 Z" fill="black" />
  </g>
</svg>
```

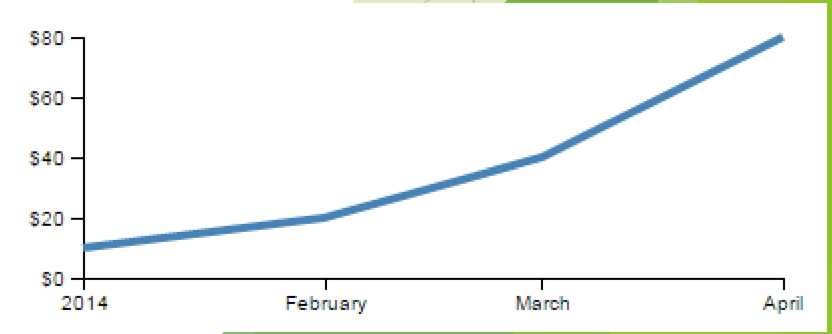


Graphing with D3.js

- In MS Excel, if you have a table of data and want to plot it, it's easy
- Using SVG to plot a graph on a web-page is a bit trickier, but made easier using D3
- To draw a graph, need to consider
  - Scales
  - Axis Labels
  - Data



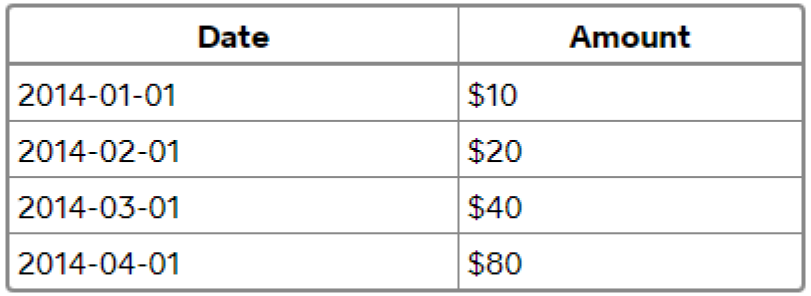
Date	Amount
2014-01-01	$10
2014-02-01	$20
2014-03-01	$40
2014-04-01	$80



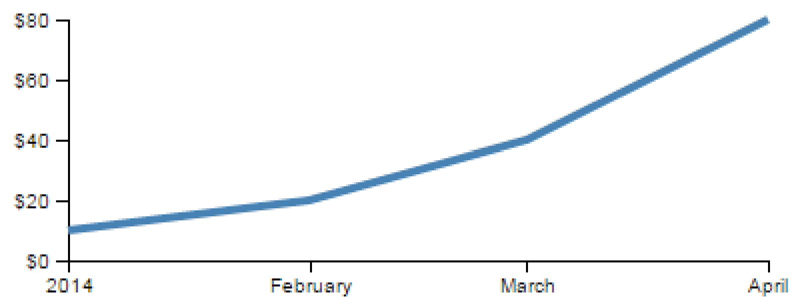
Graphing with D3.js

- D3 helps with these elements. However, D3 does this in the spirit of “automating the hard bits you already understand”, rather than making it all happen.
- The next slides illustrate how to achieve this
- In D3, our source data is always Plain Old Javascript Objects (POJOs) (also known as JSON when encoded as a string). Most often the data is a homogenous arrays of objects.

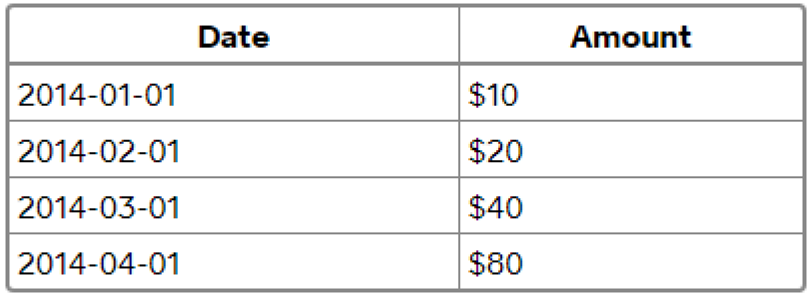
```
var numbers = [ 5, 4, 10, 1 ],
    data = [
      { date: '2014-01-01', amount: 10 },
      { date: '2014-02-01', amount: 20 },
      { date: '2014-03-01', amount: 40 },
      { date: '2014-04-01', amount: 80 }
    ];
```



Date	Amount
2014-01-01	$10
2014-02-01	$20
2014-03-01	$40
2014-04-01	$80



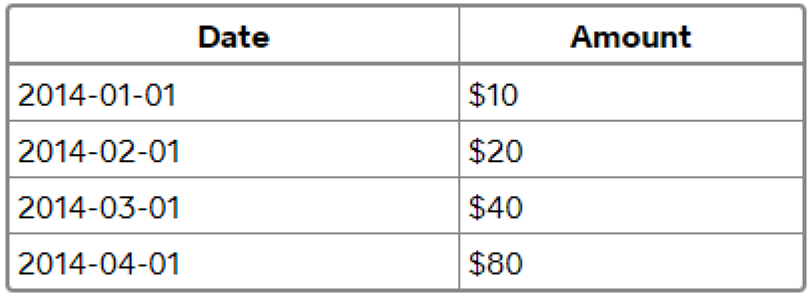
The Scale



Date	Amount
2014-01-01	$10
2014-02-01	$20
2014-03-01	$40
2014-04-01	$80

- This graph has to be “to scale”. It has to have a coordinate system
- X-axis goes from January 2014 to April 2014, and the y-axis goes from $0 to $80.
  - However, the SVG is drawn in a box that’s about 200 by 300 pixels. Dates and pixels don’t map to each other on their own, so we have to specify a mapping somehow.
- Note that the y-axis flips - SVG origin (0, 0) is in the top left, but in this graph, the origin is the bottom left. We call the chart y-up and we call SVG y-down.

The Scale



Date	Amount
2014-01-01	$10
2014-02-01	$20
2014-03-01	$40
2014-04-01	$80

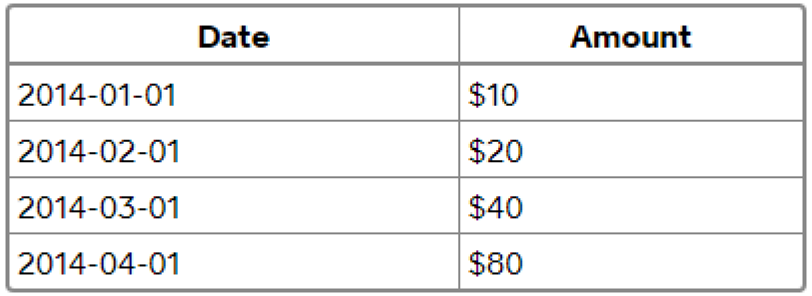
- D3 has scale objects that map values across coordinate systems.
  - There are different kinds of scales (linear, logarithmic, linear for time).
  - Scales are configured with a domain and a range
    - The domain corresponds to the data, so its units are your source units. The range is in screen space (pixels).
  - They map from the data to the appropriate part of the screen (screen space).
- Here is how we set up the y-scale for the above money example:

```
var y = d3.scaleLinear()
  .domain([0, 80]) // $0 to $80
  .range([200, 0]); // Seems backwards because SVG is y-down
```

- Or if we wanted to take advantage of the extent helper method

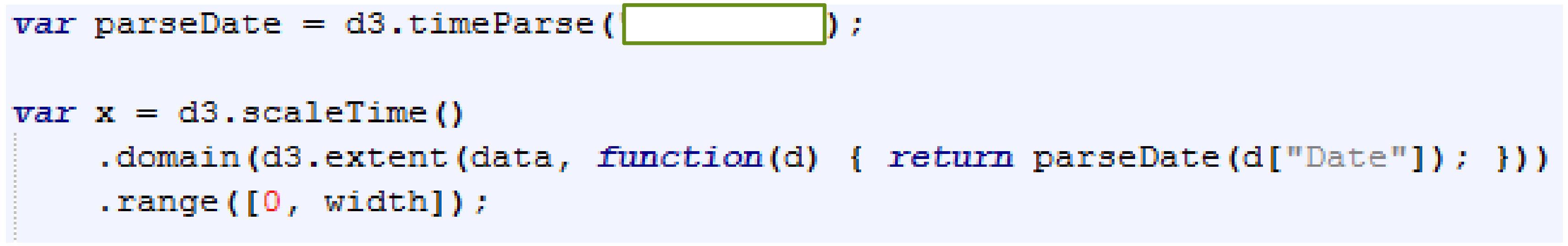
```
.domain(d3.extent(data, function(d) { return d["GBP/EUR"] })) // Extent of sterling prices
.range([height, 0]); // Seems backwards because SVG is y-down
```

The Scale



Date	Amount
2014-01-01	$10
2014-02-01	$20
2014-03-01	$40
2014-04-01	$80

- Can even do the same things with dates/time
  - Use scaleTime() instead of scaleLinear()

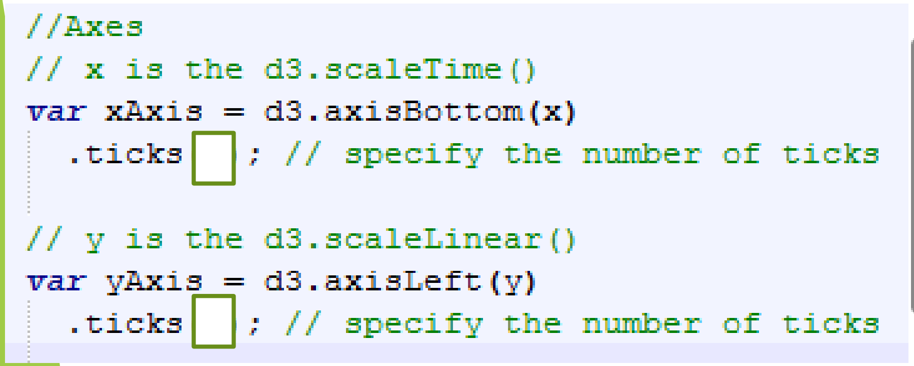


```
var parseDate = d3.timeParse(          );

var x = d3.scaleTime()
    .domain(d3.extent(data, function(d) { return parseDate(d["Date"]); }))
    .range([0, width]);
```

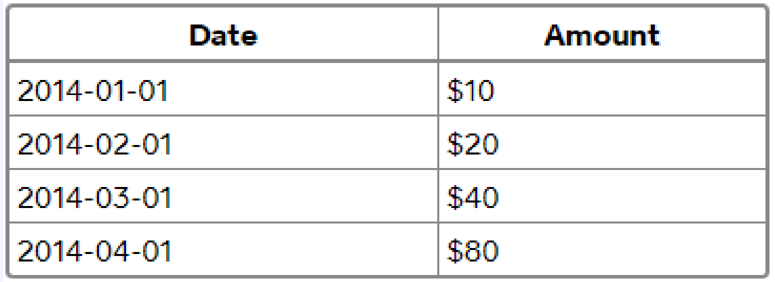
- Note the use of d3.timeParse function
  - Converts a string to a Date object that can be interpreted by the computer

Axes

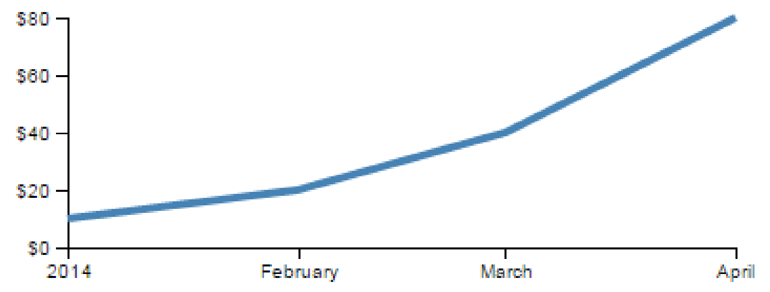
- We can read the Excel graph because it’s clearly labeled.
- Those same labels with “$20” and “February” have to be drawn to our screen somehow.
  - Also need to be formatted correctly for the data type.
- In the Excel example, there are labels and tick marks. D3 can also do this.
  - We can build an axis, and apply it to a scale.
- D3’s axes are powerful. If you use Date objects, it will label the tick marks appropriately!



```
//Axes
// x is the d3.scaleTime()
var xAxis = d3.axisBottom(x)
  .ticks   ; // specify the number of ticks

// y is the d3.scaleLinear()
var yAxis = d3.axisLeft(y)
  .ticks   ; // specify the number of ticks
```



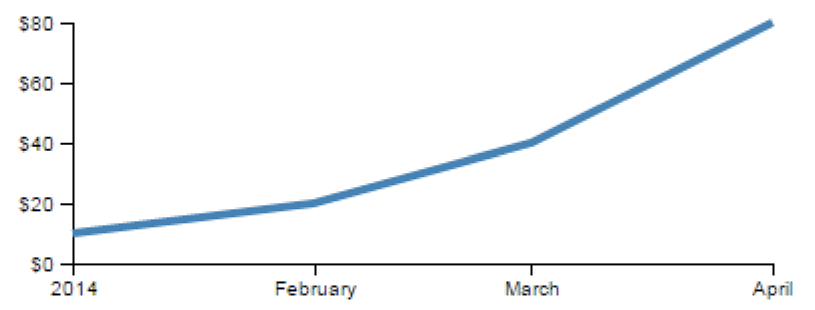
Date	Amount
2014-01-01	$10
2014-02-01	$20
2014-03-01	$40
2014-04-01	$80



Data

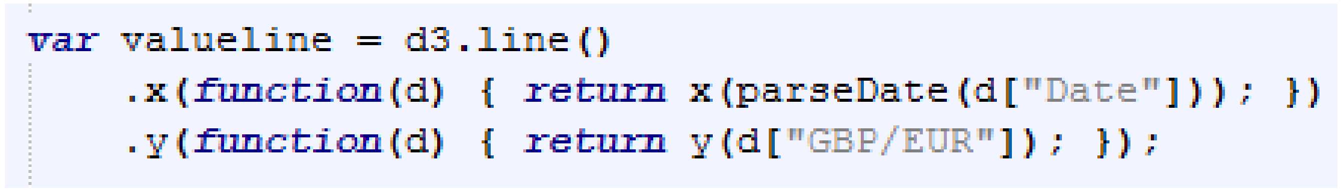
- The Excel graph is displaying the data.
  - Somehow, the 4 rows in the source table turn into 4 points on a line.
  - On top of that, the points in the line need to fit into the coordinate system
- We can intuit this, but it's critical to working with D3. We have data coming in, and we transform it to something visual.

Date	Amount
2014-01-01	$10
2014-02-01	$20
2014-03-01	$40
2014-04-01	$80



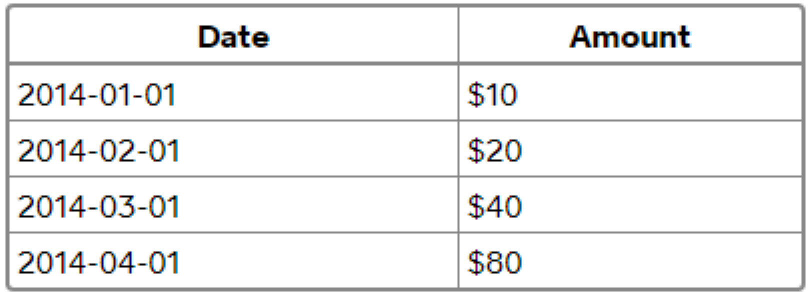
Data

- In order to draw a line, we use the d3.line() method and map the x and y coordinates for each data point using the x and y scale functions previously created:

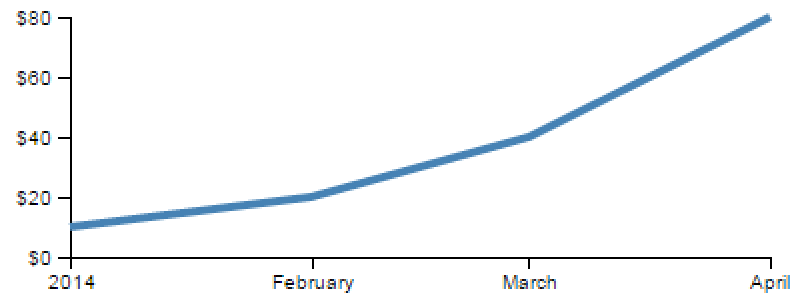


```
var valueline = d3.line()
    .x(function(d) { return x(parseDate(d["Date"])); })
    .y(function(d) { return y(d["GBP/EUR"]); });
```

- These functions will give us x and y pixel values for each of the data points
- Lots of other functions for pie/bar charts



Date	Amount
2014-01-01	$10
2014-02-01	$20
2014-03-01	$40
2014-04-01	$80



Drawing the above in SVG

- First we need to add an SVG element. We can use D3 to do this (the d3 .append method adds element to the DOM):

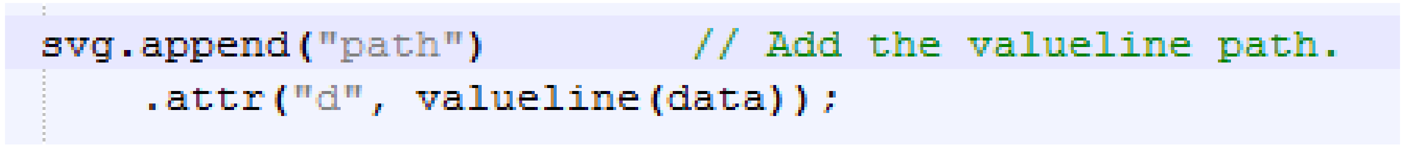
```
var svg = d3.select('body')
    .append("svg")
        .attr("width", width + margin.left + margin.right)
        .attr("height", height + margin.top + margin.bottom)

    .append("g")
        //Not necessary but adds an offset so that numeric values are visible
        .attr("transform", "translate(" + margin.left + "," + margin.top + ")");
```

- Next, add the axes:

```
svg.append('g')                  // append a <g> - group element
    .attr('class', 'x axis')
    .call(xAxis);                  // let the axis do its thing

svg.append('g')
    .attr('class', 'y axis') // specify classes
    .call(yAxis);                  // let the axis do its thing
```

Drawing the above in SVG

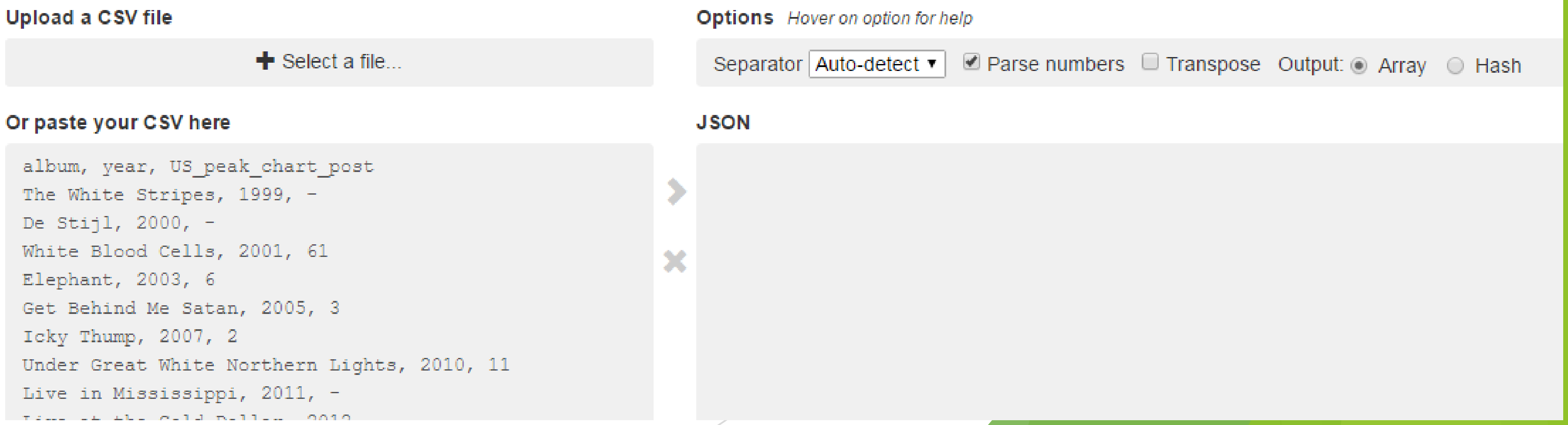
- Finally, draw the path:



```
svg.append("path")          // Add the valueline path.
    .attr("d", valueline(data));
```

This week's lab

- AWS server set up that is hosting GBP vs. EUR JSONP currency data from last 6 years: http://34.249.149.110:5000/
- Most currency data APIs cost money
  - Found a free one:
  - http://www.global-view.com/forex-trading-tools/forex-history/index.html
    - Returns CSV data
  - Needed to convert CSV to JSON
    - http://www.csvjson.com/csv2json

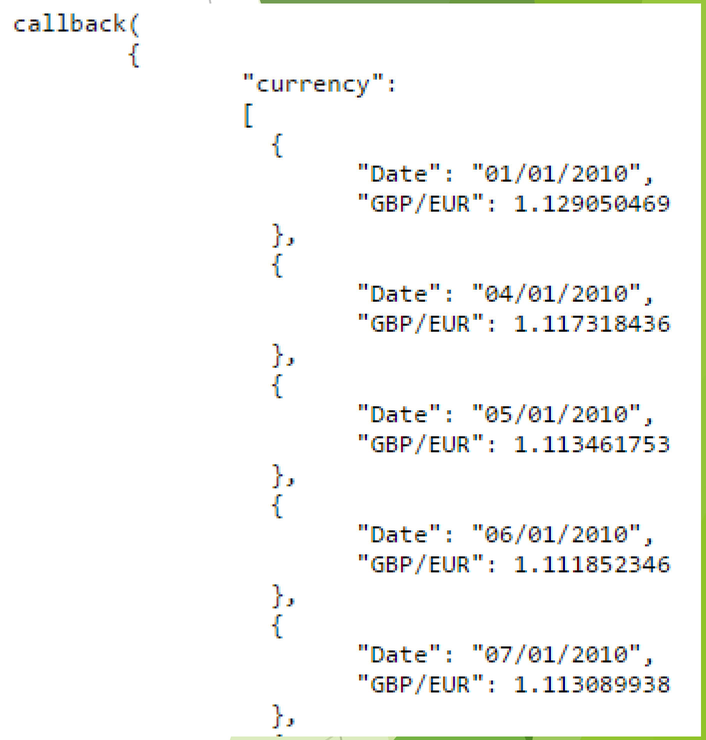


Separator Auto-detect [✓ Parse numbers] Transpose Output: [✓ Array] Hash

```
album, year, US_peak_chart_post
The White Stripes, 1999, -
De Stijl, 2000, -
White Blood Cells, 2001, 61
Elephant, 2003, 6
Get Behind Me Satan, 2005, 3
Icky Thump, 2007, 2
Under Great White Northern Lights, 2010, 11
Live in Mississippi, 2011, -
```

This week's lab

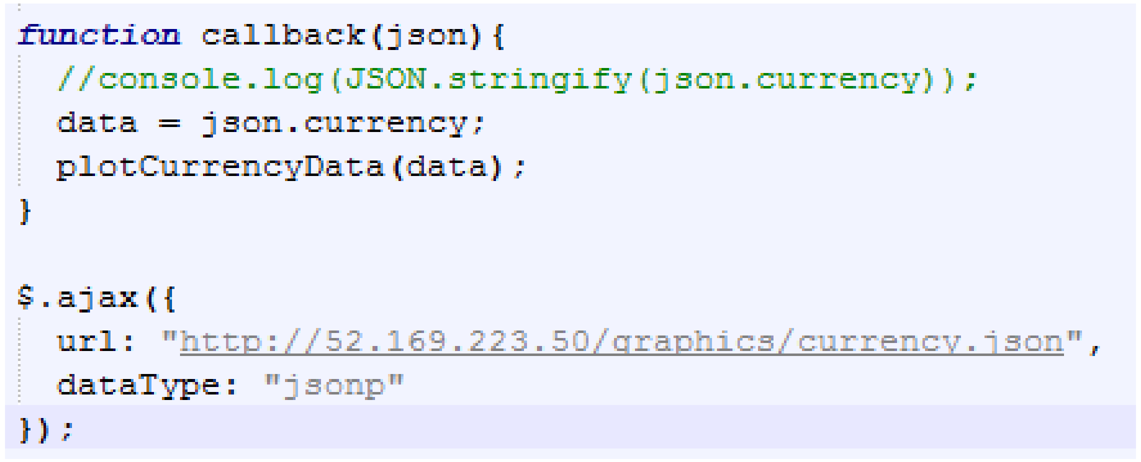
- JSONP - as in "JSON with Padding":
  - A method commonly used to bypass the cross-domain (CORS) policies in web browsers (you are not allowed to make AJAX requests to a webpage perceived to be on a different server by the browser).
  - The JSONP function invocation that gets sent to the client, and the payload that the function receives, must be agreed-upon by the client and server.
  - By convention, the server providing the JSON data offers the requesting website to name the JSONP function, typically using the name jsonp or callback as the named query parameter field name, in its request to the server,



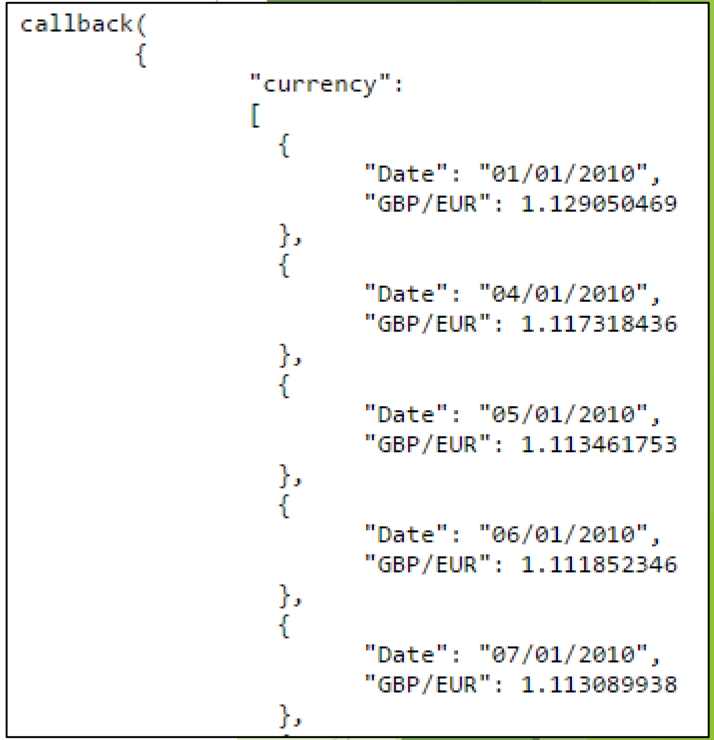
```
callback(
        {
                "currency":
                [
                  {
                          "Date": "01/01/2010",
                          "GBP/EUR": 1.129050469
                  },
                  {
                          "Date": "04/01/2010",
                          "GBP/EUR": 1.117318436
                  },
                  {
                          "Date": "05/01/2010",
                          "GBP/EUR": 1.113461753
                  },
                  {
                          "Date": "06/01/2010",
                          "GBP/EUR": 1.111852346
                  },
                  {
                          "Date": "07/01/2010",
                          "GBP/EUR": 1.113089938
                  },
```

This week's lab

- JSONP – Lots of security concerns around code-injection.
  - Used by many Web 2.0 applications such as Dojo Toolkit, Google Web Toolkit and Web services.
- Code example of how to retrieve JSON from server



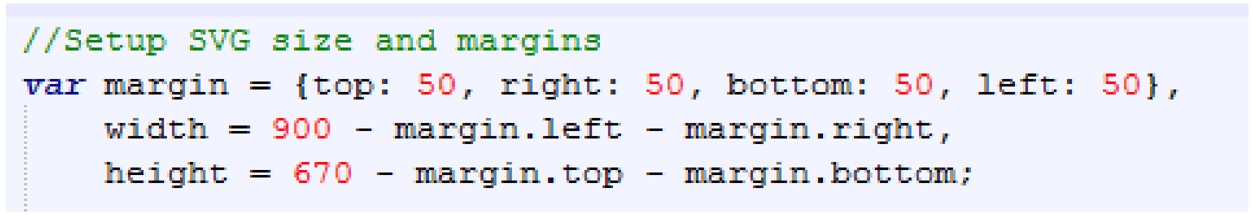
```
function callback(json){
  //console.log(JSON.stringify(json.currency));
  data = json.currency;
  plotCurrencyData(data);
}

$.ajax({
  url: "http://52.169.223.50/graphics/currency.json",
  dataType: "jsonp"
});
```



```
callback(
        {
                "currency":
                [
                  {
                        "Date": "01/01/2010",
                        "GBP/EUR": 1.129050469
                  },
                  {
                        "Date": "04/01/2010",
                        "GBP/EUR": 1.117318436
                  },
                  {
                        "Date": "05/01/2010",
                        "GBP/EUR": 1.113461753
                  },
                  {
                        "Date": "06/01/2010",
                        "GBP/EUR": 1.111852346
                  },
                  {
                        "Date": "07/01/2010",
                        "GBP/EUR": 1.113089938
                  },
```

This week's lab

- Setup SVG size and margins:



```
//Setup SVG size and margins
var margin = {top: 50, right: 50, bottom: 50, left: 50},
    width = 900 - margin.left - margin.right,
    height = 670 - margin.top - margin.bottom;
```

- Create x and y scales for GBP value and time.
  - Modify d3.timeParse to achieve this (Consult API)
- Create axis and line objects
  - Specify number of ticks in the axis
- Create SVG element
  - Add axis and line objects to the SVG element
    - Draw x-axis on bottom and y-axis on left
  - Add labels for axes, change font sizes and add a chart label
  - Advanced: Add zoom functionality (only to the x-axis)

Lab this week

- Plotting JSON data on SVG graph using D3
- Attaching zoom