Tutorial on D3.js

Klaus Mueller
Stony Brook University and SUNY Korea

SUNY Korea 2015 HOT-T-CS
Hot Topics in Computer Science

July 13 – 17, 2015
SUNY Korea, Songdo, Korea

What is D3.js?

D3 = Data Driven Documents

JavaScript library for manipulating documents based on data

- frequent tool to support *data journalism* (New York Times)

D3 helps you bring data to life using HTML, SVG, and CSS

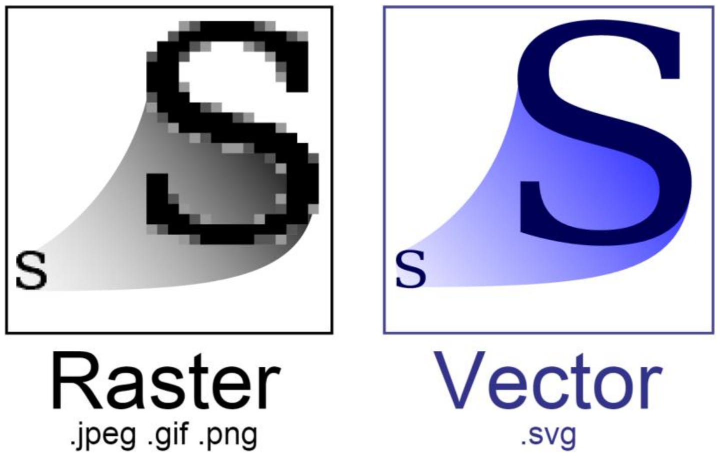
- great library to construct animated visualizations (D3 website)

Runs in any modern web browser (Chrome, Firefox, IE)

- no need to download any software
- independent of OS (Linux, Windows Mac)

Makes Use Of

HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
DOM	The Document Object Model
▪	tree structured organization of HTML objects
SVG	Scalable Vector Graphics



What You Need

A text editor

- textMate, eclipse/aptana, sublime text 2…
- need an editor with syntax highlighting. else it's easy to get lost

The d3 library

- from http://d3js.org

Data files for your code

A web server (recommended)

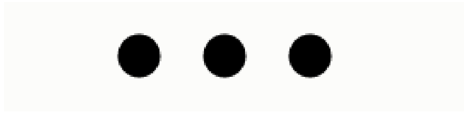
- if your visualization is reading data from files or a database (XMLHttpRequest)
- many options: EasyPHP (windows), Mac OS X Server, MAMP
- else need to specify the data in the code

A browser

- to run the code

Selections with D3

Suppose you defined three circles



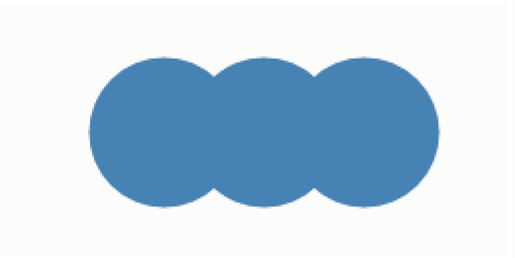
```
<svg width="720" height="120">
  <circle cx="40" cy="60" r="10"></circle>
  <circle cx="80" cy="60" r="10"></circle>
  <circle cx="120" cy="60" r="10"></circle>
</svg>
```

This will select all circles

```
var circle = d3.selectAll("circle");
```

And enlarge and fill them

```
circle.style("fill", "steelblue");
circle.attr("r", 30);
```



```
<svg width="720" height="120">
  <circle cx="40" cy="60" r="30" style="fill:steelblue;"></circle>
  <circle cx="80" cy="60" r="30" style="fill:steelblue;"></circle>
  <circle cx="120" cy="60" r="30" style="fill:steelblue;"></circle>
</svg>
```

Binding Data to Graphics

The selection.data method binds the numbers to the circles:

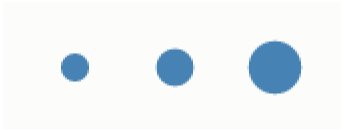
```
circle.data([32, 57, 112]);
```

Assign attributes to the bound data

- typically use the name *d* to refer to bound data

```
circle.attr("r", function(d) { return Math.sqrt(d); });
```

Will result in:

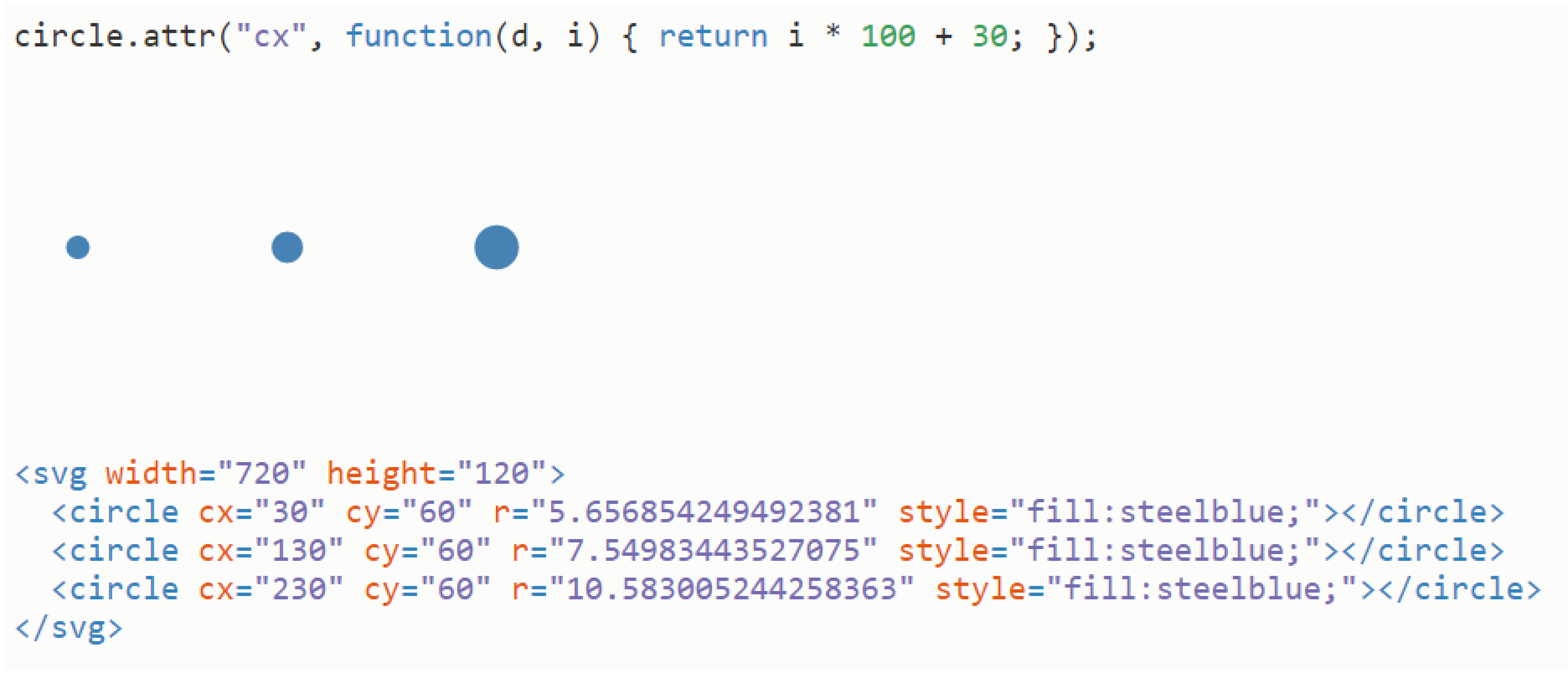


```
<svg width="720" height="120">
  <circle cx="40" cy="60" r="5.656854249492381" style="fill:steelblue;"></circle>
  <circle cx="80" cy="60" r="7.54983443527075" style="fill:steelblue;"></circle>
  <circle cx="120" cy="60" r="10.583005244258363" style="fill:steelblue;"></circle>
</svg>
```

More on Binding Data

We can use the index *i* of the data to define the graphics

Origin is the upper left corner



```
circle.attr("cx", function(d, i) { return i * 100 + 30; });
```

```
<svg width="720" height="120">
  <circle cx="30" cy="60" r="5.656854249492381" style="fill:steelblue;"></circle>
  <circle cx="130" cy="60" r="7.54983443527075" style="fill:steelblue;"></circle>
  <circle cx="230" cy="60" r="10.583005244258363" style="fill:steelblue;"></circle>
</svg>
```

Appending Graphics to Data

Suppose you have more data than graphics elements

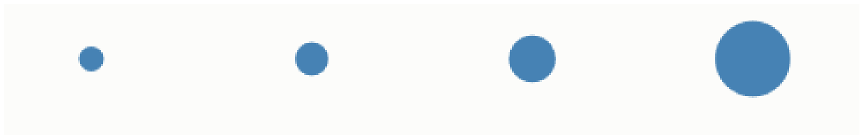
- use the enter method to add them on the fly

```
var svg = d3.select("svg");

var circle = svg.selectAll("circle")
    .data([32, 57, 112, 293]);

var circleEnter = circle.enter().append("circle");
```

- as usual, but now with 4 circles

```
circleEnter.attr("cy", 60);
circleEnter.attr("cx", function(d, i) { return i * 100 + 30; });
circleEnter.attr("r", function(d) { return Math.sqrt(d); });
```

APPENDING GRAPHICS TO DATA

(continued) we get 

```
<svg width="720" height="120">
  <circle cx="30" cy="60" r="5.656854249492381" style="fill:steelblue;"></circle>
  <circle cx="130" cy="60" r="7.54983443527075" style="fill:steelblue;"></circle>
  <circle cx="230" cy="60" r="10.583005244258363" style="fill:steelblue;"></circle>
  <circle cx="330" cy="60" r="17.11724276862369" style="fill:steelblue;"></circle>
</svg>
```

We can even begin with no circles at all:

```
svg.selectAll("circle")
    .data([32, 57, 112, 293])
  .enter().append("circle")
    .attr("cy", 60)
    .attr("cx", function(d, i) { return i * 100 + 30; })
    .attr("r", function(d) { return Math.sqrt(d); });
```

More Reading

The page where these tutorial bits came from:

http://www.lessonpaths.com/learn/i/begin-with-d3js/d3js-simplest-examples-of-d3js

Now to a more detailed, but still primitive example:

http://www.lessonpaths.com/learn/i/begin-with-d3js/d3js-simplest-examples-of-d3js

Here are some full-fledged implementations:

https://github.com/mbostock/d3/wiki/Gallery