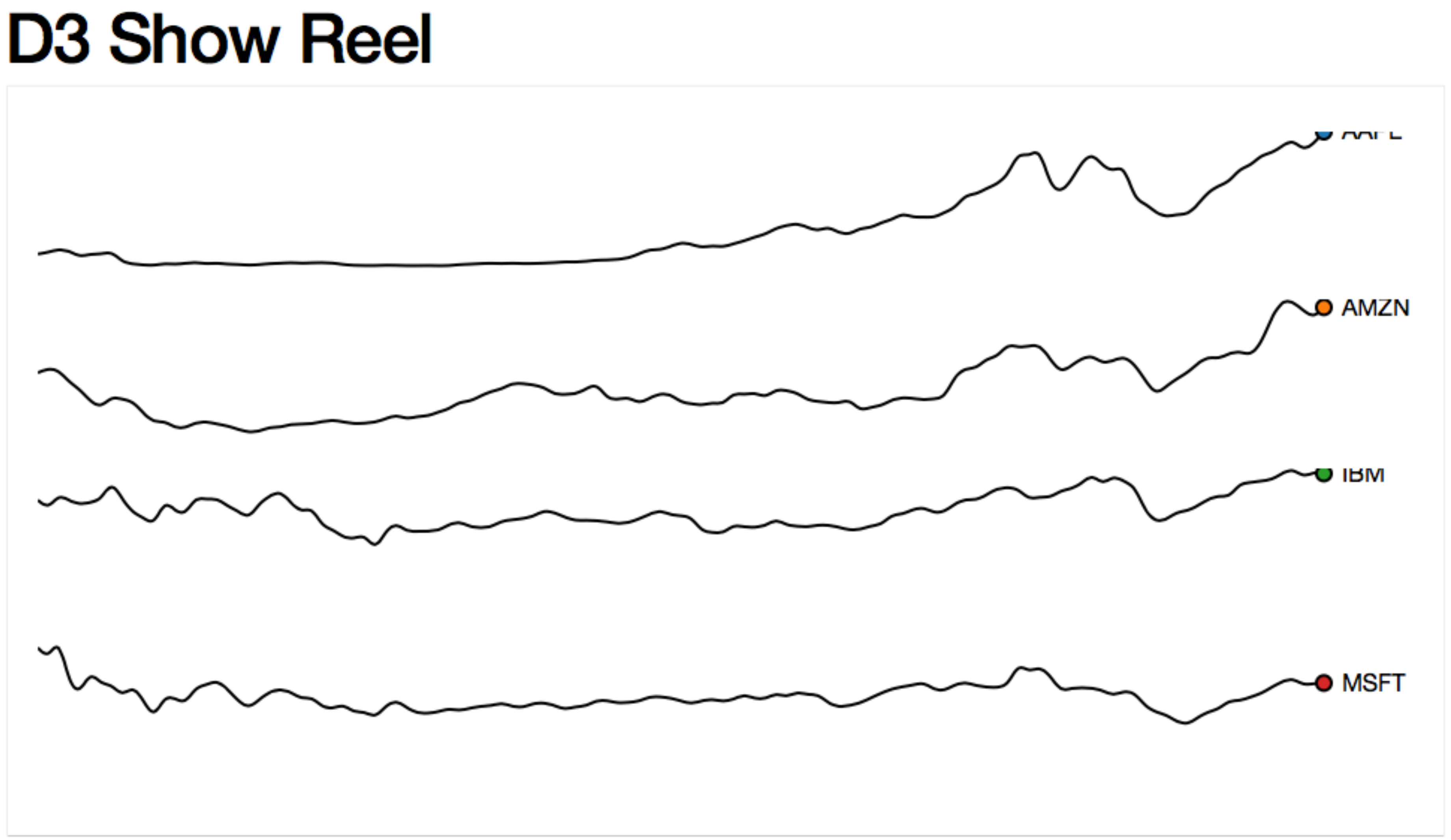
D3: The Crash Course

aka: *D3: The Early Sticking Points*
aka: *D3: Only the Beginning*

Chad Stolper
Assistant Professor
Southwestern University
(graduated from Georgia Tech CS PhD)

http://bl.ocks.org/mbostock/1256572



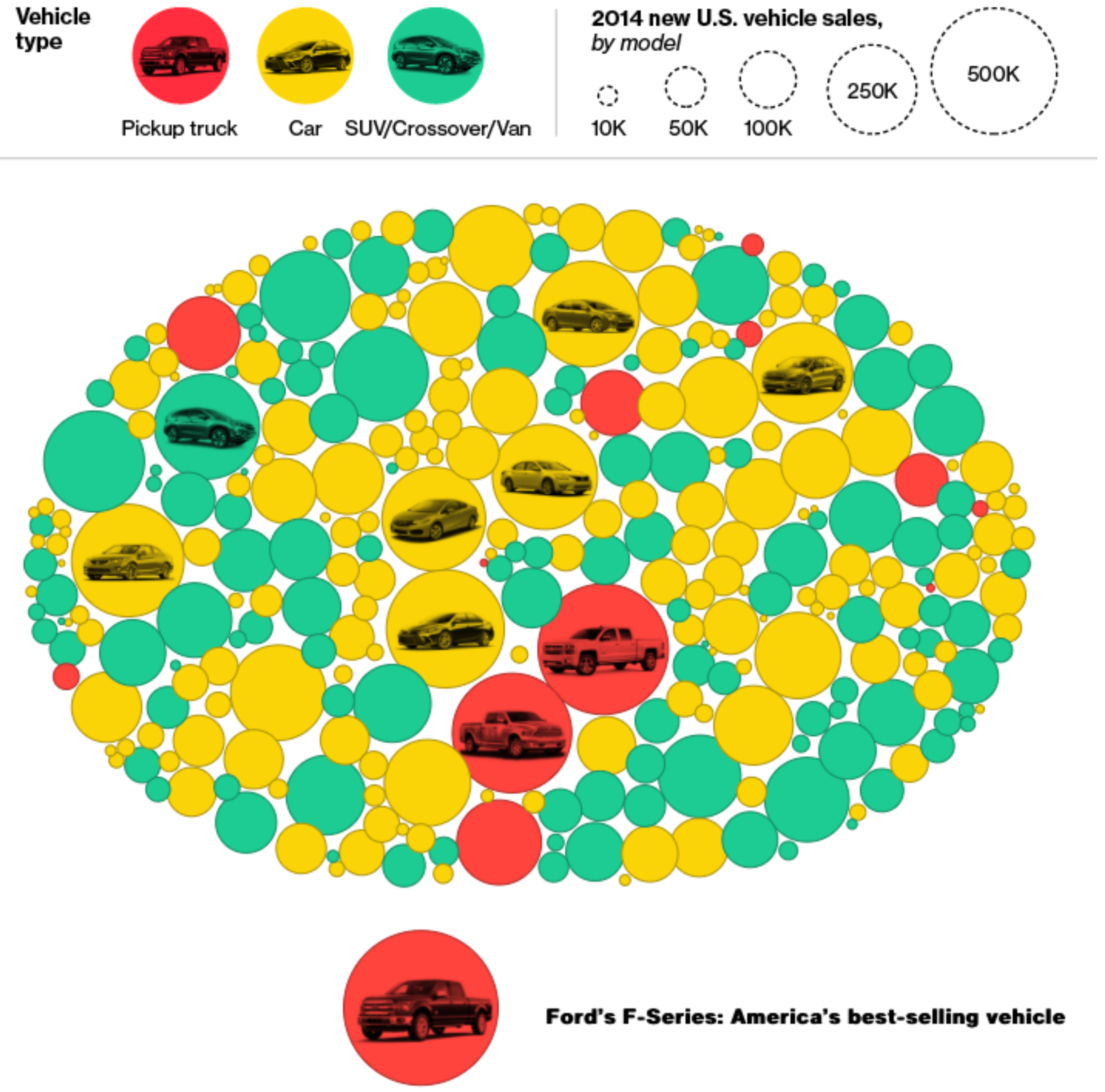
http://www.bloomberg.com/graphics/2015-auto-sales/

Pickups are king of the road.

Automakers sold more than 16.5 million new vehicles in the U.S. last year, up 5.9 percent from 2013. The most popular model, by a huge stretch, was the Ford F-Series pickup. In 2014, Americans bought 754,000 of them, making it the top-selling vehicle for the 33rd year in a row.

The F-Series trucks alone beat Volkswagen's total U.S. sales.

And Lincoln's. And Cadillac's. And Mitsubishi's. *Combined.*



BUT FIRST….

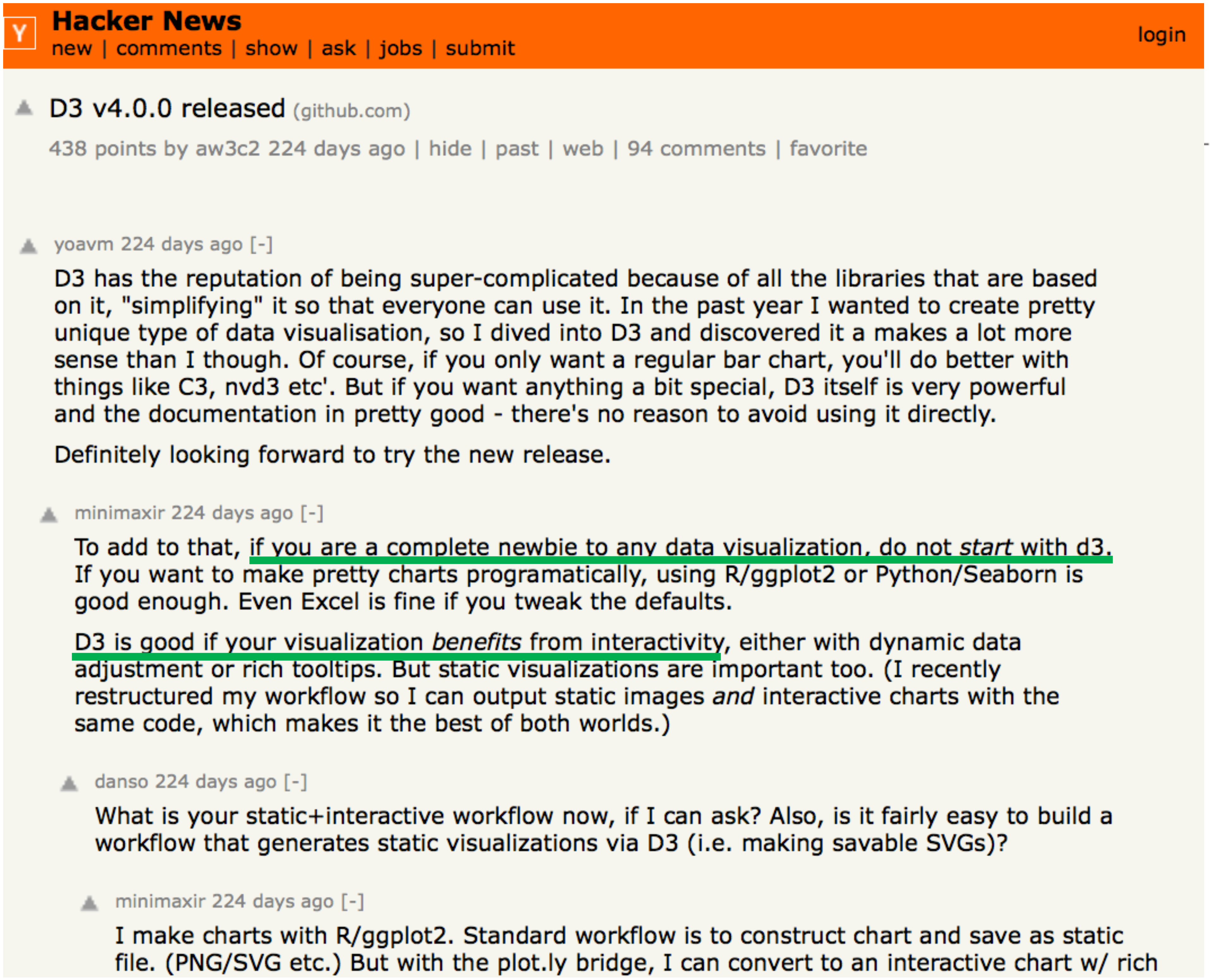
Why should you learn D3???

If you visualization/system/tool will **benefit from interactivity**.

Otherwise, use anything you want

(e.g., tableau, excel, python:seaborn, R:ggplot2, etc.)

More online discussion: https://news.ycombinator.com/item?id=11995332



Hacker News
new | comments | show | ask | jobs | submit
login

D3 v4.0.0 released (github.com)
438 points by aw3c2 224 days ago | hide | past | web | 94 comments | favorite

yoavm 224 days ago [-]

D3 has the reputation of being super-complicated because of all the libraries that are based on it, "simplifying" it so that everyone can use it. In the past year I wanted to create pretty unique type of data visualisation, so I dived into D3 and discovered it a makes a lot more sense than I though. Of course, if you only want a regular bar chart, you'll do better with things like C3, nvd3 etc'. But if you want anything a bit special, D3 itself is very powerful and the documentation in pretty good - there's no reason to avoid using it directly.

Definitely looking forward to try the new release.

minimaxir 224 days ago [-]

To add to that, if you are a complete newbie to any data visualization, do not *start* with d3. If you want to make pretty charts programatically, using R/ggplot2 or Python/Seaborn is good enough. Even Excel is fine if you tweak the defaults.

D3 is good if your visualization *benefits* from interactivity, either with dynamic data adjustment or rich tooltips. But static visualizations are important too. (I recently restructured my workflow so I can output static images *and* interactive charts with the same code, which makes it the best of both worlds.)

danso 224 days ago [-]

What is your static+interactive workflow now, if I can ask? Also, is it fairly easy to build a workflow that generates static visualizations via D3 (i.e. making savable SVGs)?

minimaxir 224 days ago [-]

I make charts with R/ggplot2. Standard workflow is to construct chart and save as static file. (PNG/SVG etc.) But with the plot.ly bridge, I can convert to an interactive chart w/ rich

This lectures and HW2 is about D3 ver.3

- Ver4 is the latest, but has "breaking" changes.
- Most D3 examples/tutorials are still using v3
- Ver4 vs ver3: https://iros.github.io/d3-v4-whats-new/#1
- Upgrading Ver3 code to ver4 code: https://keithpblog.wordpress.com/2016/07/31/upgrading-d3-from-v3-to-v4/

Chrome Inspector and Console

- Open the webpage
- Right-click on anything
- Click “inspect”
- Open the console too, so you can see the error messages

Starting a Local Web Server

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

Necessary for Chrome, not for Safari or Firefox

(This is a security measure: to prevent reading from your file systems)

- Python 2.x
  - python -m SimpleHTTPServer 8000
- Python 3.x
  - python –m http.server 8000
- http://localhost:8000

If you're new to JavaScript…

prepare for a lot of…

confusion (wat??)

and hair pulling

I'm serious.

https://siouxfallsradioadvertisingdotcom.files.wordpress.com/2011/11/mad-man-pulling-hair-out.jpg

If you’re new to Javascript…

https://www.destroyallsoftware.com/talks/wat

(starting 1:20)

Javascript 101

- All variables are global, unless declared using **var**
  - x = 300 (global)
  - var x = 300 (local)
- Semicolons are optional
- “text” is the same as ‘text’
- JS arrays and objects are almost exactly the same syntax as python’s lists [ ] and dicts { }
- object.key is the same as object[‘key’]
- Print to the console using console.log( )

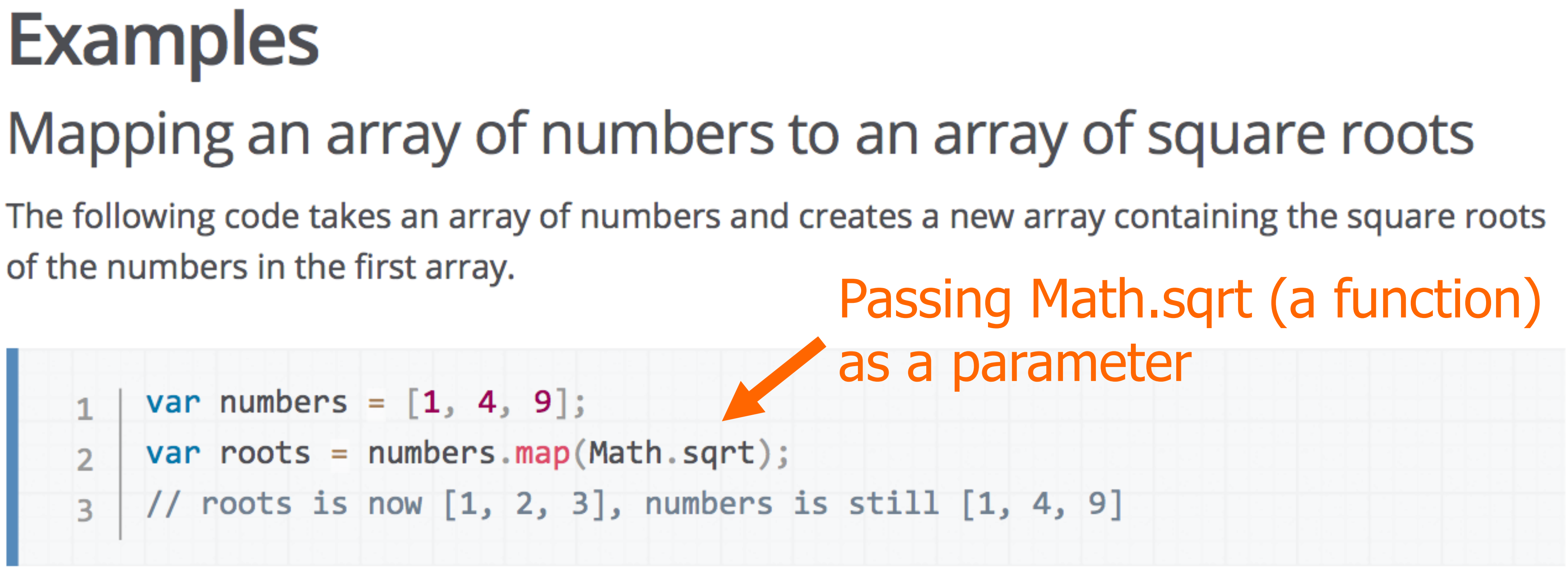
Javascript 102: Functional Programming

- Javascript supports **functional programming**
  - Functions are themselves objects
  - Functions can be stored as variables
  - Functions can be passed as parameters
    - As in HW1: http://alignedleft.com/tutorials/d3/making-a-bar-chart
- D3 uses these abilities extensively!

Some people say javascript is a “multi-paradigm” programming language. http://stackoverflow.com/questions/3962604/is-javascript-a-functional-programming-language

What does that mean?

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map



Examples

Mapping an array of numbers to an array of square roots

The following code takes an array of numbers and creates a new array containing the square roots of the numbers in the first array.

```
var numbers = [1, 4, 9];
var roots = numbers.map(Math.sqrt);
// roots is now [1, 2, 3], numbers is still [1, 4, 9]
```

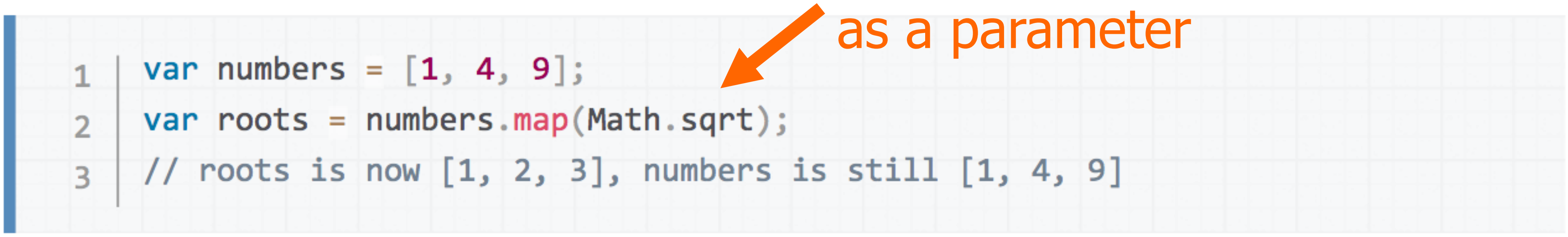
Passing Math.sqrt (a function) as a parameter

Array.map( )

- Used for applying a function to each element of an array

- The function provided (Math.sqrt) takes one value as input (e.g., 9) and output another value (e.g., 3).

Passing Math.sqrt (a function) as a parameter



```javascript
var numbers = [1, 4, 9];
var roots = numbers.map(Math.sqrt);
// roots is now [1, 2, 3], numbers is still [1, 4, 9]
```

MDN – the BEST Javascript reference

- Mozilla Developer Network
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference
- (Easier: google "<command> mdn")

Method Chaining

- "Syntactic Sugar" paradigm where each method returns the object that it was called on

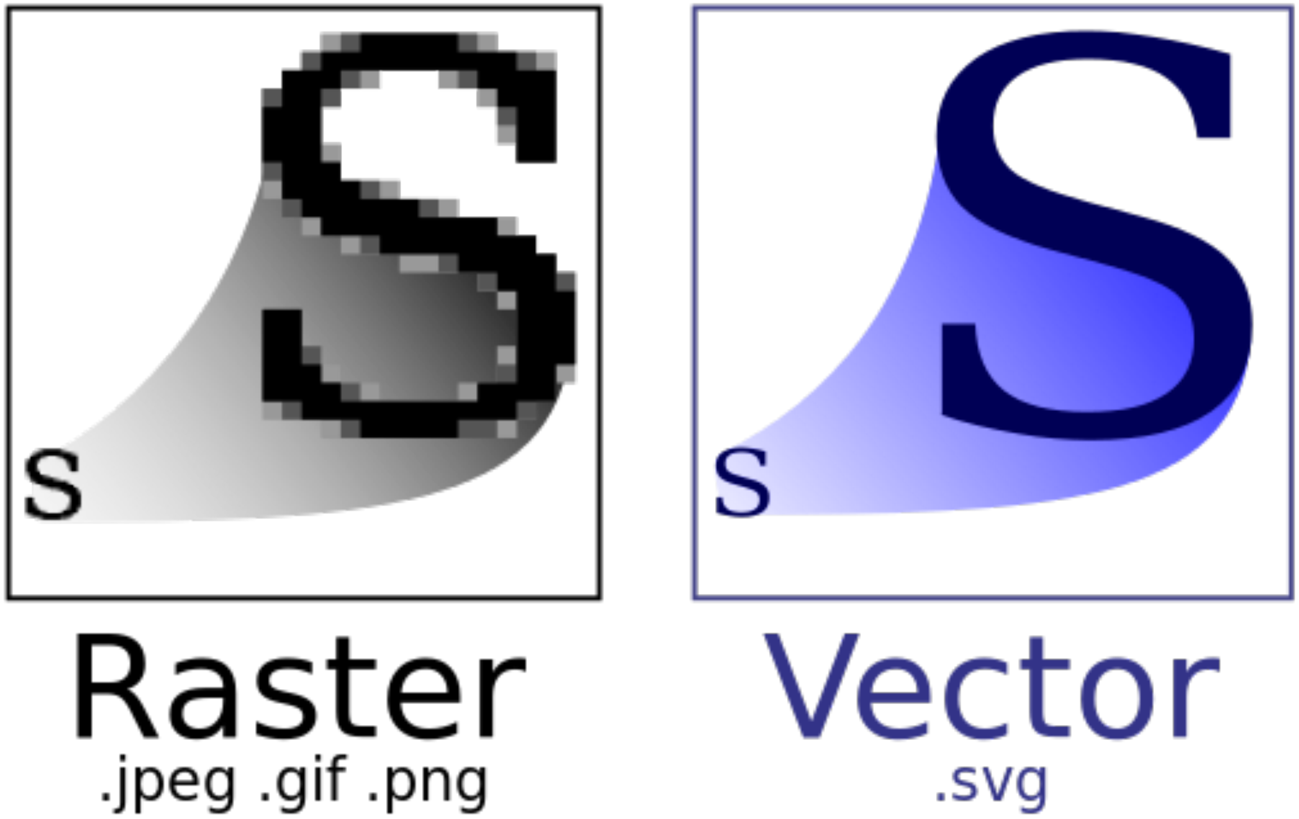
```
group
     .attr("x",5)
     .attr("y",5); //returns group
```

is the same as

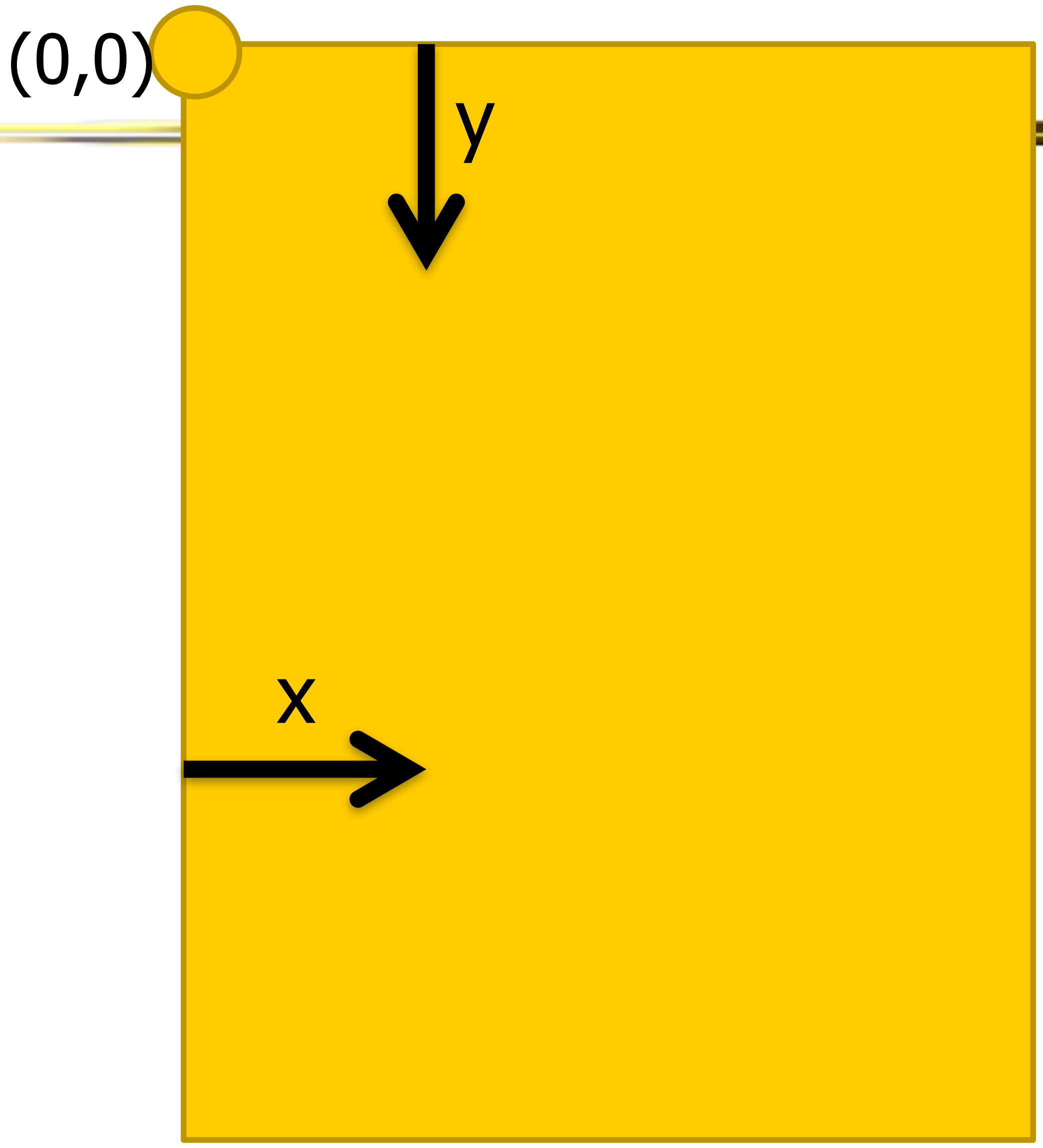
```
group.attr("x",5) //returns group
group.attr("y",5) //returns group
```

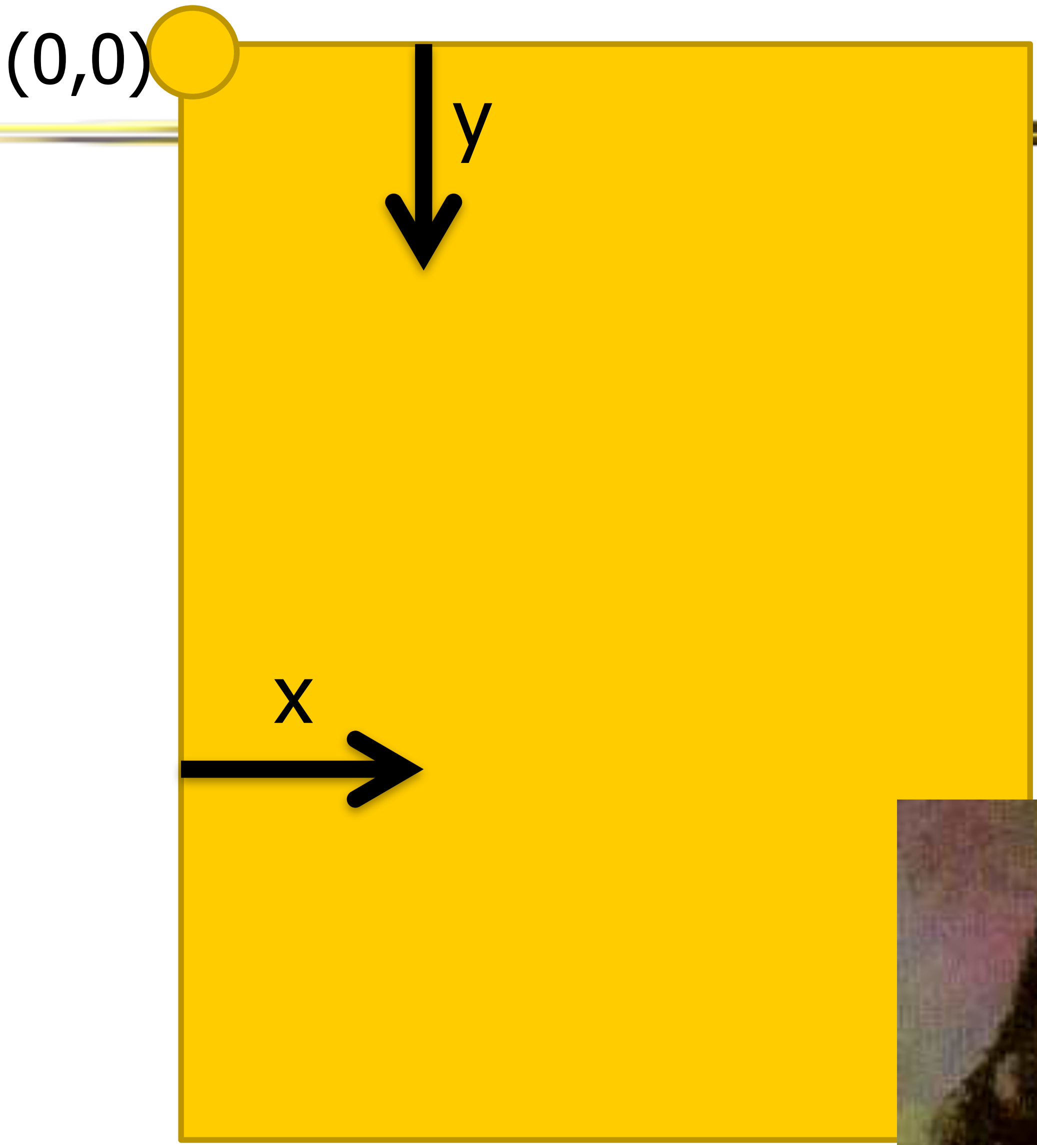
SVG BASICS

SVG = Scalable Vector Graphics



https://en.wikipedia.org/wiki/Scalable_Vector_Graphics


(0,0)
y
x



http://smg.photobucket.com/user/Pavan2099/media/RvB/Descart-weeping.png.html

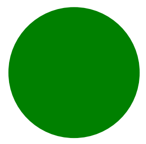
SVG Basics

SVG -> XML Vector Graphics

(<u>S</u>calable <u>V</u>ector <u>G</u>raphics)

SVG Basics

- XML Vector Graphics
  - Tags with Attributes
  - `<circle r=5 fill="green"></circle>`



- W3C Standard
  - http://www.w3.org/TR/SVG/
- Supported by all the major browsers

SVG Basics

- <svg>
- <circle>
- <rect>
- <path>
- <g>

SVG Basics

- <svg>
- <circle>
- <rect>
- <path>
- <g>
- <text> (after I’ve talked about D3)

<svg> element

- Overarching drawing board
- (optional) Attributes:
  - width
  - height

```
<body>
  <div id="vis">
  </div>
</body>
```

- Create with
  - `d3.select("#vis").append("svg")`

<svg> element

- Overarching canvas
- (optional) Attributes:
  - width
  - height
- Create with
  - `d3.select("#vis").append("svg")`

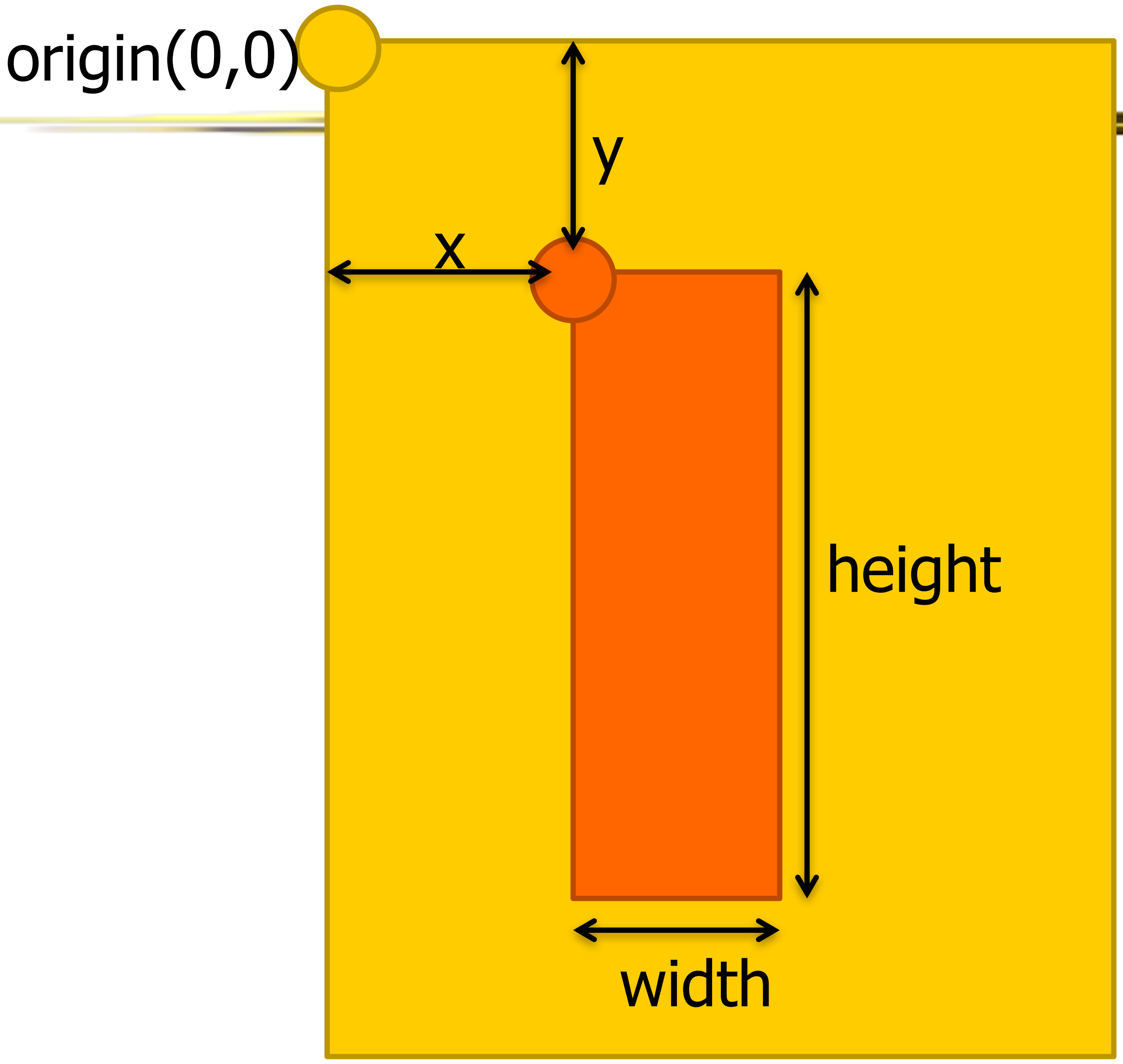
```
<body>
  <div id="vis">
      <svg></svg>
  </div>
</body>
```

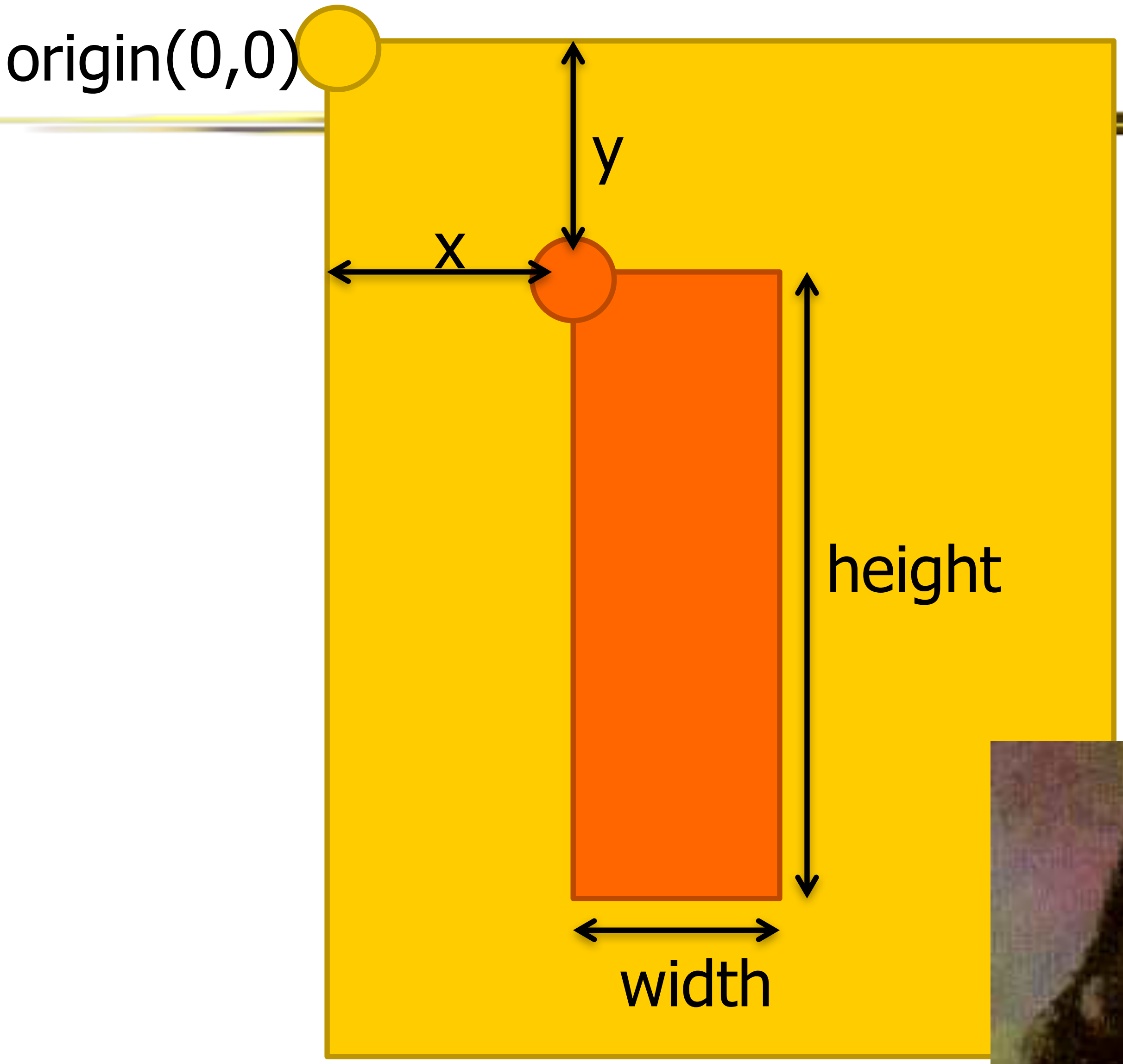
<circle> element

- Attributes:
  - cx (relative to the LEFT of the container)
  - cy (relative to the TOP of the container)
  - r (radius)
- (optional) Attributes:
  - fill (color)
  - stroke (the color of the stroke)
  - stroke-width (the width of the stroke)
- Create with
  - `.append("circle")`

<rect> element

- Attributes:
  - x (relative to the LEFT of the container)
  - y (relative to the TOP of the container)
  - width (cannot be negative)
  - height (cannot be negative)
- (optional) Attributes:
  - fill (color)
  - stroke (the color of the stroke)
  - stroke-width (the width of the stroke)
- Create with
  - `.append("rect")`


origin(0,0)
y
x
height
width



http://smg.photobucket.com/user/Pavan2099/media/RvB/Descart-weeping.png.html

Rather than positioning each element, what if we want to position (or style) a group of elements?

<g> element

- Generic container (Group) element

- Attributes
  - transform
  - (fill,stroke,etc.)
- Create with:
  - `var group = vis.append(“g”)`
- Add things to the group with:
  - `group.append(“circle”)`
  - `group.append(“rect”)`
  - `group.append(“text”)`

CSS Selectors Reference

- #vis → <tag id="vis">
- circle → <circle>
- .canary → <tag class="canary">
- [color="blue"] → <tag color="blue">
- And many more ways
  - http://www.w3schools.com/cssref/css_selectors.asp

- And any combinations…
  - AND
    - circle.canary → <circle class="canary">
  - OR
    - circle, .canary → <circle> <circle class="canary"> <tag class="canary">

AND NOW D3…

Mike Bostock and Jeff Heer @ Stanford

2009- Protovis

Mike Bostock and Jeff Heer @ Stanford

2009- Protovis

Mike Bostock and Jeff Heer @ Stanford

2009- Protovis

2011- D3.js

Univ. of Washington

Mike Bostock and Jeff Heer @ ~~Stanford~~

2009- Protovis

2011- D3.js

New York Times Univ. of Washington

Mike Bostock and Jeff Heer @ ~~Stanford~~

2009- Protovis

2011- D3.js

D3

- Grand Reductionist Statements

- Loading Data
- Enter-Update-Exit Paradigm
- Scales
- Axes
- Layouts
- Transitions and Interaction

- Where to go from here

D3.js in a Nutshell

D3 is a really powerful for-loop
with a ton of useful helper functions

D3

Declarative, domain-specific specification language for manipulating the DOM

Importing D3

```
<html >
    <head>
        <script src='lib/d3.js’ charset=‘utf-8’></script>
        <script src='js/project.js'></script>
    </head>
    <body>
        <div id=“vis”></div>
    </body>
</html>
```

Importing D3

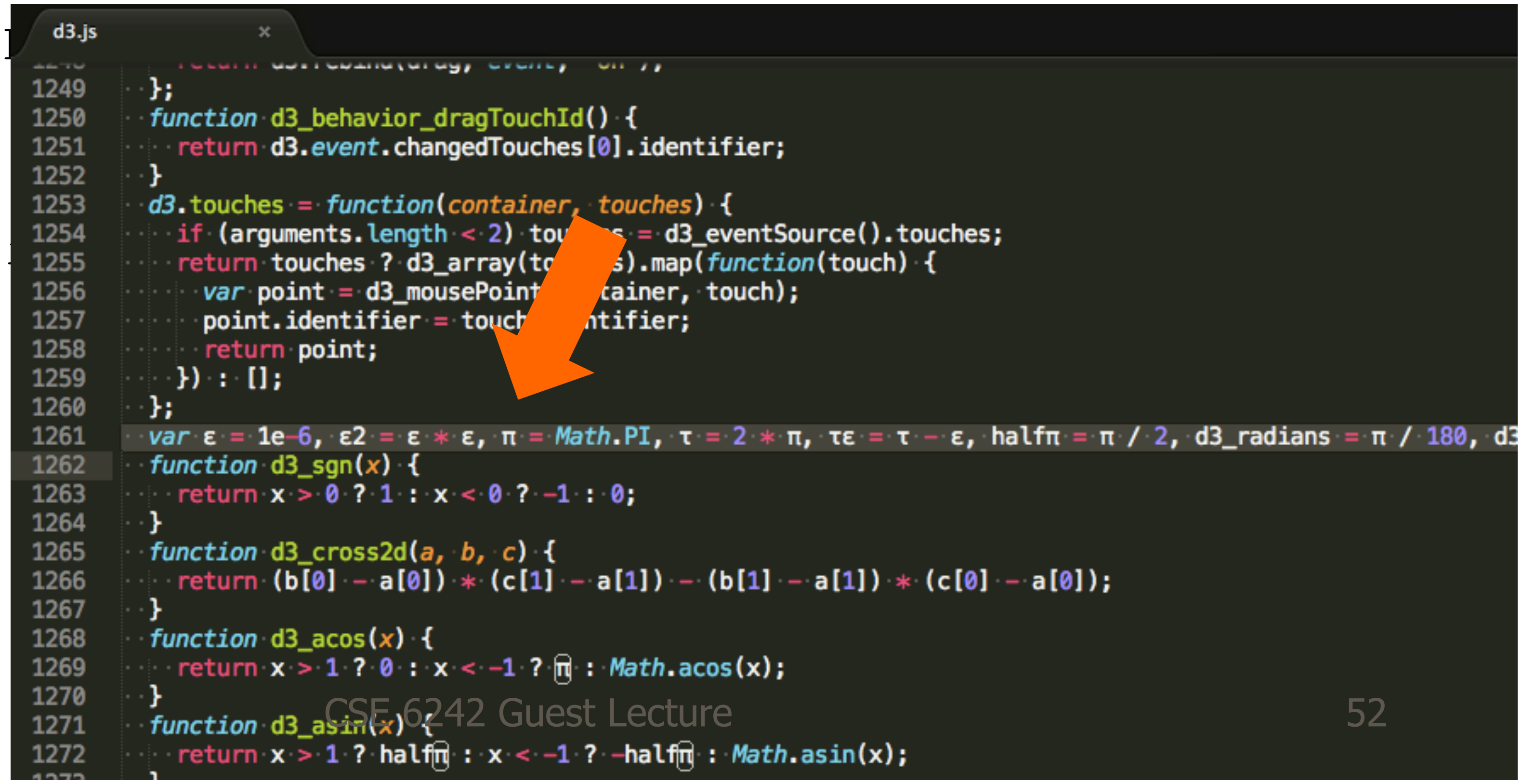
```
<html >
    <head>
        <script src='lib/d3.js’ charset=‘utf-8’></script>
        <script src='js/project.js'></script>
    </head>
    <body>
        <div id=“vis”></div>
    </body>
</html>
```

Importing D3

```
<html >
    <head>
        <script src='lib/d3.js' charset='utf-8'></script>
        <script src='js/project.js'></script>
    </head>
    <body>
        <div id="vis"></div>
    </body>
</html>
```

Importing D3

```
<html >
    <head>
        <script src='lib/d3.js' charset='utf-8'></script>
        <scri
    </head>
    <body>
        <div
    </body>
</html>
```



d3.js

```
};
function d3_behavior_dragTouchId() {
  return d3.event.changedTouches[0].identifier;
}
d3.touches = function(container, touches) {
  if (arguments.length < 2) tou      s = d3_eventSource().touches;
  return touches ? d3_array(to      s).map(function(touch) {
    var point = d3_mousePoint      tainer, touch);
    point.identifier = touch      ntifier;
    return point;
  }) : [];
};
var ε = 1e-6, ε2 = ε * ε, π = Math.PI, τ = 2 * π, τε = τ - ε, halfπ = π / 2, d3_radians = π / 180, d3
function d3_sgn(x) {
  return x > 0 ? 1 : x < 0 ? -1 : 0;
}
function d3_cross2d(a, b, c) {
  return (b[0] - a[0]) * (c[1] - a[1]) - (b[1] - a[1]) * (c[0] - a[0]);
}
function d3_acos(x) {
  return x > 1 ? 0 : x < -1 ? π : Math.acos(x);
}
function d3_asin(x) {
  return x > 1 ? halfπ : x < -1 ? -halfπ : Math.asin(x);
```

Importing D3

```
<html >
    <head>
        <script src='lib/d3.js’ charset=‘utf-8’></script>
        <script src='js/project.js'></script>
    </head>
    <body>
        <div id=“vis”></div>
    </body>
</html>
```

Assigning the Canvas to a Variable

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

```
<body>
     <div id="vis"><svg></svg></div>
</body>
```

Loading Data

- `d3.csv(fileloc,callback)`
- `d3.tsv(fileloc,callback)`
- `d3.json(fileloc,callback)`

- fileloc: string file location
  - "data/datafile.csv"
- callback: `function(rawdata){ }`

rawdata from a CSV file

```
[
  {
    'name': 'Adam',
    'school': 'GT',
    'age': '18'
  },
  {
    'name': 'Barbara',
    'school': 'Emory',
    'age': '22'
  },
  {
    'name': 'Calvin',
    'school': 'GSU',
    'age': '30'
  }
]
```

name	school	age
Adam	GT	18
Barbara	Emory	22
Calvin	GSU	30

Problem

```
[
  {
    'name': 'Adam',
    'school': 'GT',
    'age': '18'
  },
  {
    'name': 'Barbara',
    'school': 'Emory',
    'age': '22'
  },
  {
    'name': 'Calvin',
    'school': 'GSU',
    'age': '30'
  }
]
```

- Ages are Strings!
- They should be ints!
- We can fix that:

```
for(var d: data){
        d = data[d]
        d.age = +d.age
 }
```

Problem

```
[
  {
    'name': 'Adam',
    'school': 'GT',
    'age': '18'
  },
  {
    'name': 'Barbara',
    'school': 'Emory',
    'age': '22'
  },
  {
    'name': 'Calvin',
    'school': 'GSU',
    'age': '30'
  }
]
```

- Ages are Strings!
- They should be ints!
- We can fix that:

```
for(var d: data){
     d = data[d]
     d.age = +d.age
}
```

WAT

http://stackoverflow.com/questions/24473733/importing-a-csv-into-d3-cant-convert-strings-to-numbers

rawdata from a CSV file

```
[
  {
    'name': 'Adam',
    'school': 'GT',
    'age': 18
  },
  {
    'name': 'Barbara',
    'school': 'Emory',
    'age': 22
  },
  {
    'name': 'Calvin',
    'school': 'GSU',
    'age': 30
  }
]
```

name	school	age
Adam	GT	18
Barbara	Emory	22
Calvin	GSU	30

rawdata from a CSV file

```
[
  {
    'name': 'Adam',
    'school': 'GT',
    'age': 18
  },
  {
    'name': 'Barbara',
    'school': 'Emory',
    'age': 22
  },
  {
    'name': 'Calvin',
    'school': 'GSU',
    'age': 30
  }
]
```

name	school	age
Adam	GT	18
Barbara	Emory	22
Calvin	GSU	30

Ok, so let's map this data to visual elements!

D3

Declarative, domain-specific specification language for manipulating the DOM

Define a **template** for each element

D3 draws one element for each data point

Enter-Update-Exit

- The *most* critical facet of how D3 works

- If you remember nothing else from today, remember this...
- "Enter-Update-Exit"
- "Enter-Update-Exit"
- "Enter-Update-Exit"

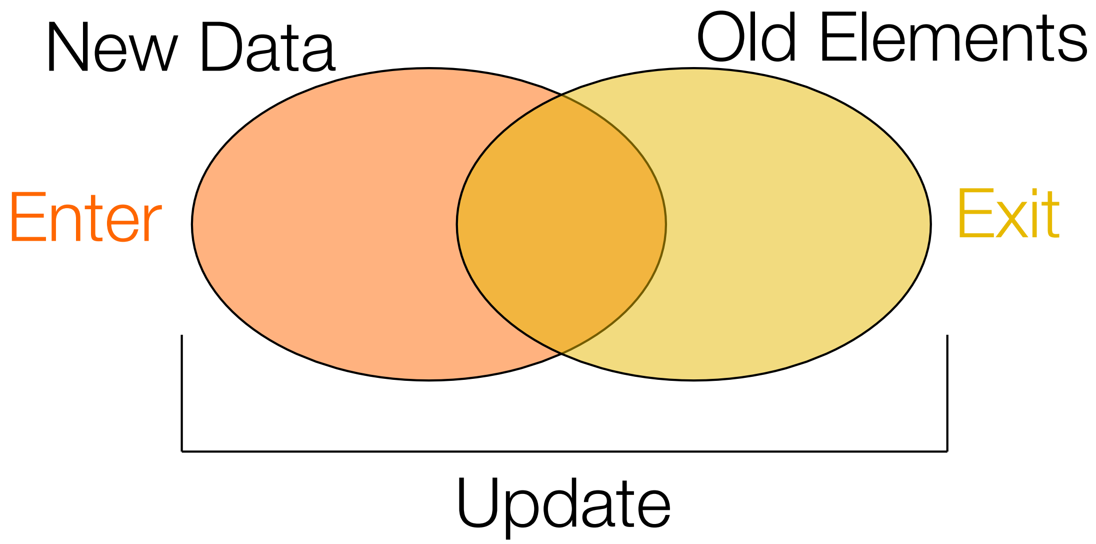
Enter-Update-Exit

Pattern:

- Select a “group” of “elements” (e.g., circles)
- Assign **data** to the **group**
- **Enter**: Create new elements for data points not associated with any elements yet (and set constant or initial attribute values)
- **Update**: Set the attributes of all the elements based on the data
- **Exit**: Remove elements that don’t have data anymore

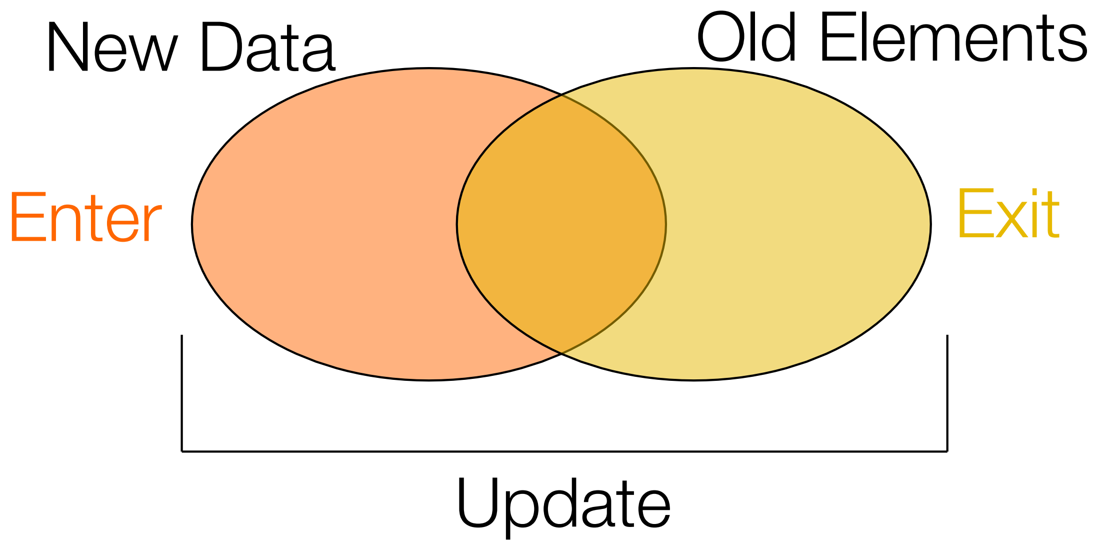
.enter( ) and .exit( )

- .data([1,2,3,4])
  - Enter: [1,2,3,4]
  - Update: [1,2,3,4]
  - Exit: [ ]
- .data ([1,2,3,4,5,6])
  - Enter: [5,6]
  - Update: [1,2,3,4,5,6]
  - Exit: [ ]
- .data ([1,2,3])
  - Enter: [ ]
  - Update: ???
  - Exit: [4,5,6]



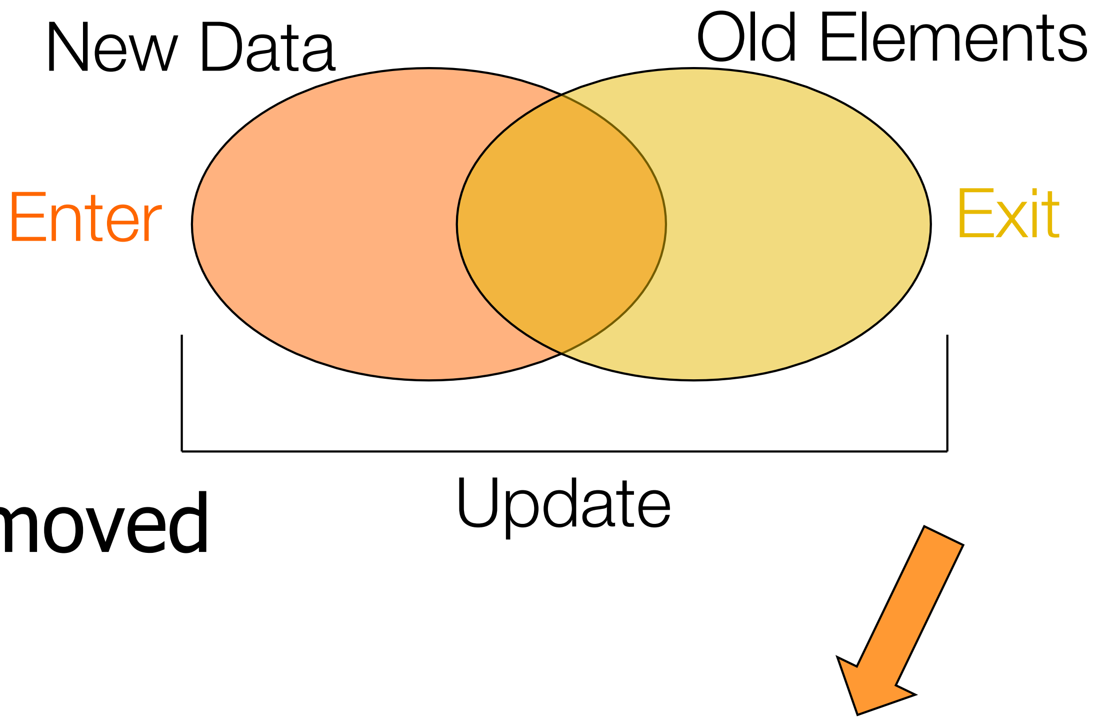
.enter( ) and .exit( )

- .data([1,2,3,4])
  - Enter: [1,2,3,4]
  - Update: [1,2,3,4]
  - Exit: [ ]
- .data ([1,2,3,4,5,6])
  - Enter: [5,6]
  - Update: [1,2,3,4,5,6]
  - Exit: [ ]
- .data ([1,2,3])
  - Enter: [ ]
  - Update: [1,2,3,4,5,6]
  - Exit: [4,5,6]



.enter( ) and .exit( )

- .enter( )
  - New data points
- .exit( )
  - Elements to be removed



- .enter() and .exit() only exist when .data() has been called

Can be hard to grok:

You can select groups of elements that DON'T EXIST YET

http://bost.ocks.org/mike/join/

Still confused?

Enter, Update, Exit
An Introduction to D3.js, The Web's Most Popular Visualization Toolkit

Excellent interactive demo to explain enter-update-exit:
http://niceone.org/examples/d3-selections/

Full tutorial:
https://medium.com/@c_behrens/enter-update-exit-6cafc6014c36#.dqwkermdb

Data Key Functions

- .data(rawdata) defaults to assuming that the index of the point is the key
- .data(rawdata, function(d,i){ }) allows you to set a key functions
- e.g.
  - `.data(rawdata, function(d,i){ return d.id; })`
  - `.data(rawdata, function(d,i){ return d.name; })`

E-U-E Pattern Template

```
var group = vis.selectAll(“rect”)
      .data(rawdata) //rawdata must be an array!
group.enter( ).append(“rect”) //ENTER!
      .attr( )
      .style( )
group //UPDATE!
      .attr( )
      .style( )
group.exit( ).remove( ) //EXIT!
```

WARNING!!!!

E-U-E Pattern Template

```
var group = vis.selectAll(“rect”)
      .data(rawdata) //rawdata must be an array!
group.enter( ).append(“rect”) //ENTER!
      .attr( )
      .style( )
group //UPDATE!
      .attr( )
      .style( )
group.exit( ).remove( ) //EXIT!
```

Many online examples

E-U-E Pattern Template

```
var group = vis.selectAll("rect")
      .data(rawdata) //rawdata must be an array!
~~group~~.enter( ).append("rect") //ENTER!
      .attr( )
      .style( )
group //UPDATE!
      .attr( )
      .style( )
group.exit( ).remove( ) //EXIT!
```

Many online examples drop the variable name before .enter()

E-U-E Pattern Template

```
var group = vis.selectAll("rect")
      .data(rawdata) //rawdata must be an array!
group.enter( ).append("rect") //ENTER!
      .attr( )
      .style( )
group //UPDATE!
      .attr( )
      .style( )
group.exit( ).remove( ) //EXIT!
```

Many online examples drop the variable name before .enter()

I *highly* recommend you don't!

.attr( )

- The Attribute Method
- Sets attributes such as x, y, width, height, and fill

- Technical details:
  - `group.attr(“x”, 5)`
  - `<rect x=“5”></rect>`

.attr( ) and Functional Programming

Input

```
[ {size: 10}, {size: 8}, {size: 12.2} ]
```

We want 3 rectangles:

```
   <rect height=“10” x=“5”></rect>
   <rect height=“8” x=“10”></rect>
   <rect height=“12.2” x=“15”></rect>
```

```
.attr(“height”, function(d,i){ return d.size })
   d: the data point
.attr(“x”, function(d,i){ return (i+1)*5; })
   i: the index of the data point
```

<text> elements

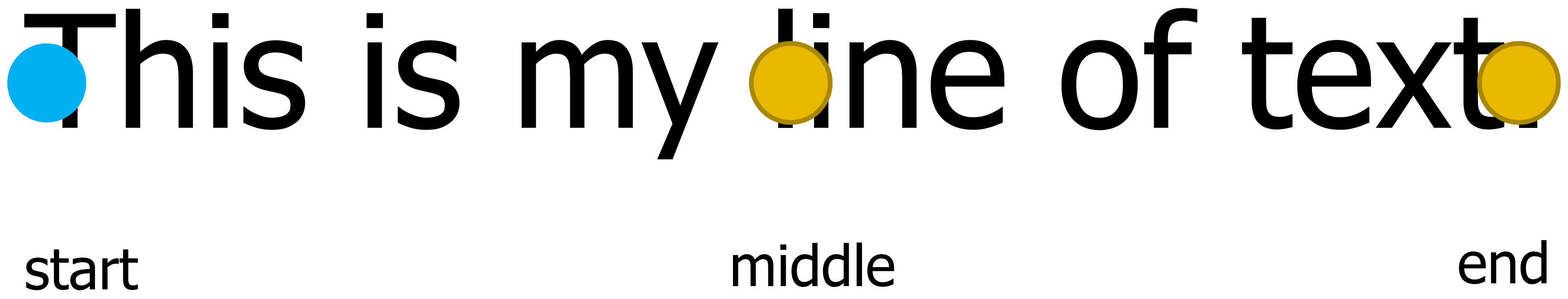
- I’m going to apologize in advance here for the lousy job the W3C did with the <text> definition.
- You’re going to have to just either memorize these things or keep referring back to http://www.w3c.org/TR/SVG/text.html (first Google hit for “svg text”) like I do.

<text> elements

- Extra Method in D3
  - `.text("Your Text Goes Here")`
  - `<tag>Your Text Goes Here</tag>`
- Attributes
  - x
  - y
- Styles
  - text-anchor: start, middle, end
  - dominant-baseline: [nothing], hanging, middle

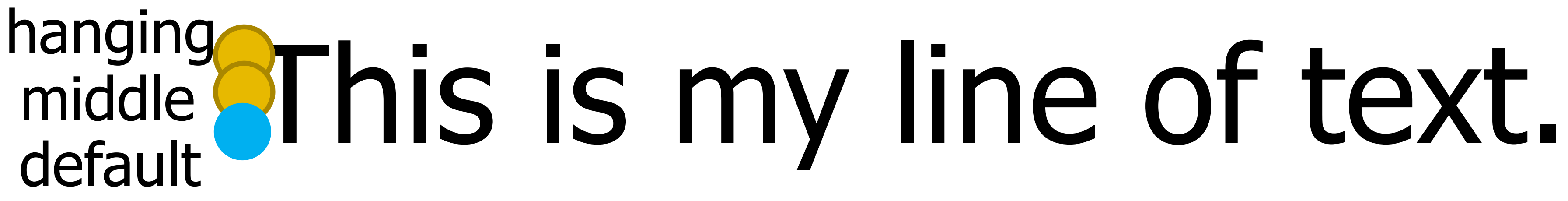
text-anchor style

Where is (0,0)?

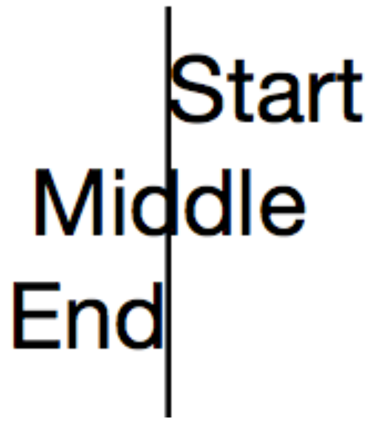


dominant-baseline style

Where is (0,0)?



<text> example



```
<text x="50" y="20"
      style="text-anchor: start">
    Start
</text>
<text x="50" y="40"
      style="text-anchor: middle">
    Middle
</text>
<text x="50" y="60"
      style="text-anchor: end">
    End
</text>
```

http://tutorials.jenkov.com/svg/text-element.html

The .style() Function

Like attr, but for the style attribute

- Inline CSS styling

```
.style("prop1","val1")
.style("prop2","val2")
.style("prop3", function(d,i){ })

<ele style="prop1: val1; prop2: val2;">
```

<text> example

```
group.append("svg:text")
      .text(function(d){return d.name})
      .attr("x", function(d,i){return i*5})
      .attr("y", function(d,i){return height;})
      .style("dominant-baseline","hanging")
      .style("text-anchor", "middle")
```

Need to remember what to use .style and when to use .attr

What if you have
two different types of circles?

Classing

- CSS Classes
  - Any number of classes per element
  - Select using ".classname"

```
blue = vis.selectAll("circle.bluecircle")
      .data(bluedata, function(d){return d.id;})
blue.enter( ).append("svg:circle")
      .classed("bluecircle", "true")
vis.selectAll(".bluecircle").attr("fill","blue")
```

Scales

(e.g., sizing a circle based on data value)

```
.attr("height", function(d){ return d; })
```

can blow up really quickly…

Scales

- D3 has many types of scales
- I am only going to cover two:
  - Linear Scales
  - Ordinal Scales

Linear Scales

```
var xscale = d3.scale.linear( )
    .domain( [min, max] )
    .range( [minOut, maxOut] )

group.attr(“x”, function(d,i){
    return xscale(d.size);
})
```

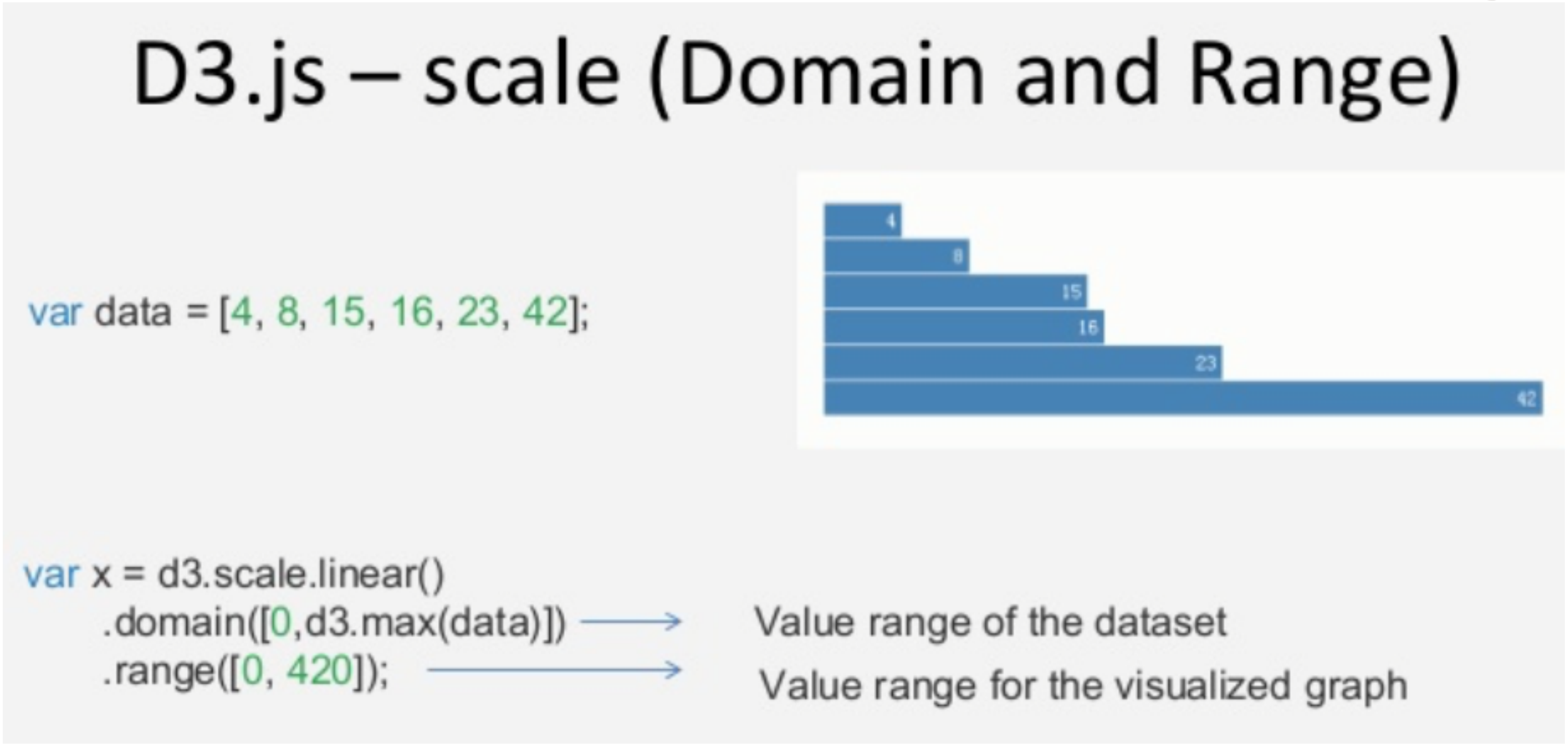
Min and Max

But how do you figure out the min and max for the domain?

D3

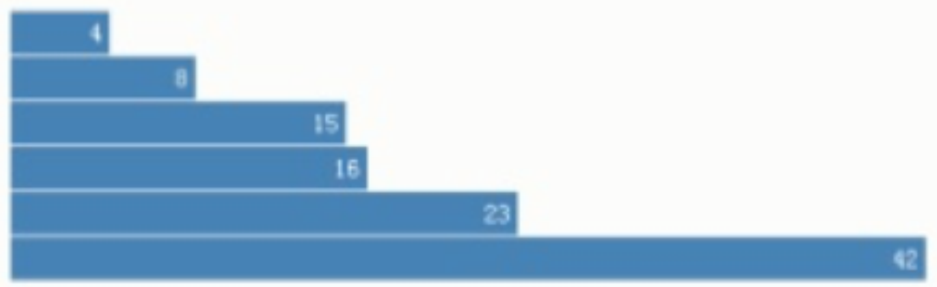
A really powerful for-loop with a ton of useful helper functions

Domain & Range



D3.js – scale (Domain and Range)

```
var data = [4, 8, 15, 16, 23, 42];
```



```
var x = d3.scale.linear()
      .domain([0,d3.max(data)])   →   Value range of the dataset
      .range([0, 420]);           →   Value range for the visualized graph
```

http://image.slidesharecdn.com/d3-140708145630-phpapp02/95/d3-17-638.jpg?cb=1404831405

Min and Max

- d3.min([ ]) → number
- d3.max([ ]) → number
- d3.extent([ ]) → [number,number]

Min and Max

- d3.min([ ]) → number
- d3.max([ ]) → number
- d3.extent([ ]) → [number,number]

An optional ***accessor*** function may be specified, which is equivalent to calling ***array.map(accessor)*** before computing the maximum value.

```
d3.max(
   data.map( function(d){ return d.age; })
) // returns the maximum age
```

https://github.com/d3/d3-3.x-api-reference/blob/master/Arrays.md

```
var maxAge = d3.max(
   data.map( function(d){ return d.age; })
) // returns the maximum age

var yscale = d3.scale.linear( )
     .domain( [0, maxAge] )
     .range( [0, 100] )
```

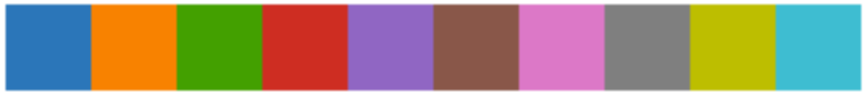
Ordinal Scales

- D3 has built-in color scales!
  - (And they're easy!)
- var colorscale = d3.scale.category10( )
- Also available are:
  - category20( )
  - category20b( )
  - category20c( )
  - (and even a few more)

~~Ordinal~~ Categorical Scales

- D3 has built-in color scales!
  - (And they're easy!)

- var colorscale = d3.scale.category10( )



- Also available are:
  - category20( )
  - category20b( )
  - category20c( )
  - (and even a few more)

Think carefully before using a rainbow palette for ordinal data!

http://www.mathworks.com/tagteam/81137_92238v00_RainbowColorMap_57312.pdf

~~Ordinal~~ Categorical Scales

- [{type:‘Bird’},{type:‘Rodent’},{type:‘Bird’}]
- var colorscale = d3.scale.category10( )
- .attr(“fill”,function(d,i){

```
      return colorscale(d.type)
 }
```

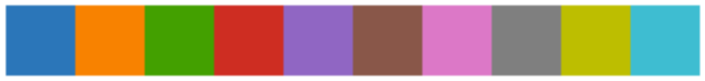
  –

  –

  –

~~Ordinal~~ Categorical Scales

- `[ {type:'Bird'},{type:'Rodent'},{type:'Bird'} ]`
- `var colorscale = d3.scale.category10( )`
-
```
.attr("fill",function(d,i){
    return colorscale(d.type)
 }
```



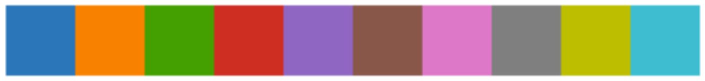
Bird Blue

  –
  –
  –

~~Ordinal~~ Categorical Scales

- `[ {type:‘Bird’},{type:‘Rodent’},{type:‘Bird’} ]`
- `var colorscale = d3.scale.category10( )`
- `.attr(“fill”,function(d,i){`

```
    return colorscale(d.type)
 }
```

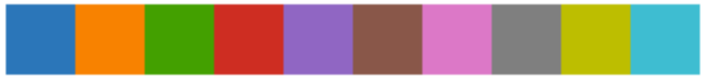


Bird Blue

  – `<rect fill=“blue”></rect>`
  –
  –

~~Ordinal~~ Categorical Scales

- `[ {type:'Bird'},{type:'Rodent'},{type:'Bird'} ]`
- `var colorscale = d3.scale.category10( )`
-
```
.attr("fill",function(d,i){
    return colorscale(d.type)
 }
```
  - `<rect fill="blue"></rect>`



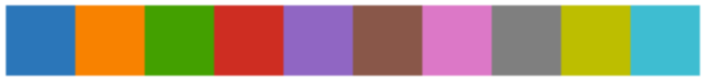
Bird	Blue
Rodent	Orange

~~Ordinal~~ Categorical Scales

- `[ {type:'Bird'},{type:'Rodent'},{type:'Bird'} ]`
- `var colorscale = d3.scale.category10( )`
- `.attr("fill",function(d,i){`

```
      return colorscale(d.type)
  }
```

  - `<rect fill="blue"></rect>`
  - `<rect fill="orange"></rect>`
  -



Bird	Blue
Rodent	Orange

~~Ordinal~~ Categorical Scales

- [{type:'Bird'},{type:'Rodent'},{type:'Bird'}]
- var colorscale = d3.scale.category10( )
- .attr("fill",function(d,i){

  ```
      return colorscale(d.type)
  }
  ```

  - <rect fill="blue"></rect>
  - <rect fill="orange"></rect>
  - <rect fill="blue"></rect>

Bird	Blue
Rodent	Orange

D3 also has *visual* helper-functions

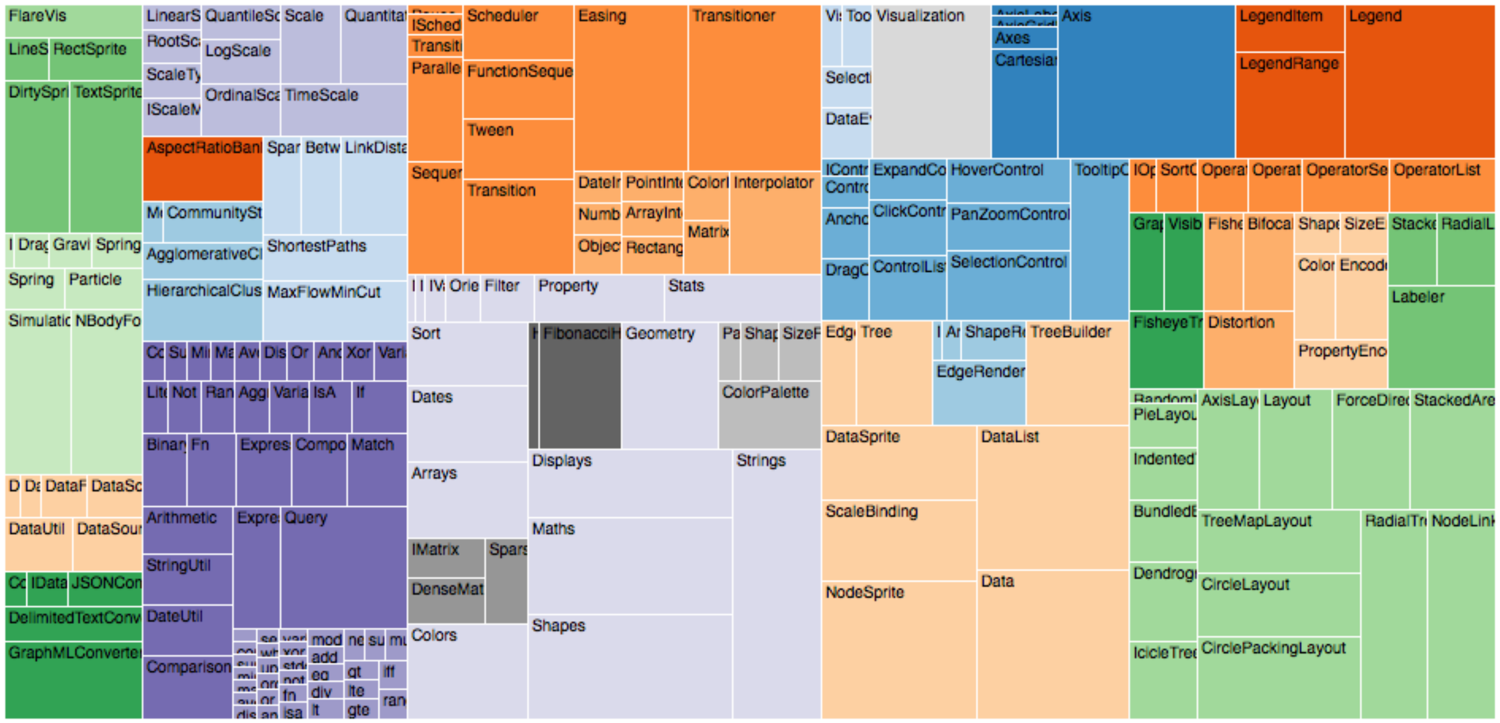
Axes

```
yaxisglyph = vis.append(“g”)

yaxis = d3.svg.axis( )
   .scale( yscale ) //must be a numerical scale
   .orient( ‘left’ ) //or ‘right’,‘top’, or ‘bottom’
   .ticks( 6 ) //number of ticks, default is 10
yaxisglyph.call(yaxis)
```

D3 even has some entire techniques built in (e.g., treemap)…

http://bl.ocks.org/mbostock/4063582



Chad Stolper

What if the data is changing?

E-U-E Pattern Template

```
var group = vis.selectAll(“rect”)
      .data(rawdata) //rawdata must be an array!
group.enter( ).append(“svg:rect”) //ENTER!
      .attr( )
      .attr( )
group //UPDATE!
      .attr( )
      .attr( )
group.exit( ).remove( ) //EXIT!
```

E-U-E Pattern Template

```
function redraw(rawdata){
   var group = vis.selectAll(“rect”)
      .data(rawdata) //rawdata must be an array!
   group.enter( ).append(“svg:rect”) //ENTER!
      .attr( )
      .attr( )
   group //UPDATE!
      .attr( )
      .attr( )
   group.exit( ).remove( ) //EXIT!
}
```

E-U-E Pattern Template

```
function redraw(rawdata){
   var group = vis.selectAll("rect")
      .data(rawdata) //rawdata must be an array!
   group.enter( ).append("svg:rect") //ENTER!
      .attr( )
      .attr( )
   group.transition( ) //UPDATE!
      .attr( )
      .attr( )
   group.exit( ).remove( ) //EXIT!
}
```

Transitions

- CSS3 transitions with D3 are *magical!*
- D3 interpolates values for you…

Transitions

```
rect.attr(“height”, 0)
rect.transition( )
       .delay( 500 ) //can be a function of data
       .duration(200) //can be a function of data
       .attr(“height”, 5) //can be a function of data
       .style(“fill”,”green”) //can be a function of data
```

So transitions allow a vis to be dynamic…

But they're not really interactive…

Interaction

The on( ) Method

.on( )

```
rect.on (“click”, function(d){
   d.color = “blue”;
   redraw( rawdata )
})
```

HTML Events

- click
- mouseover
- mouseenter
- mouseout
- etc.

.on( )

```
rect.on (“click”, function(d){
   d.color = “blue”;
   redraw( rawdata )
})
```

d is the data point backing the element clicked on

HTML Events

- click
- mouseover
- mouseenter
- mouseout
- etc.

Where to get learn more…

- http://d3js.org/
  - Tons of examples and basics.
- https://github.com/mbostock/d3/wiki/API-Reference
  - Official D3 documentation. Extremely well done.
- https://github.com/mbostock/d3/wiki/Tutorials
  - List of seemingly ALL the tutorials online
- The Google/StackOverflow combination
  - (my personal favorite)

When You're Bored...

http://www.koalastothemax.com/

Thanks!

chadstolper@gatech.edu

Good Luck!

chadstolper@gatech.edu

Questions?

chadstolper@gatech.edu