Vis Programming Tutorial

CS 7450 - Information Visualization
Sep. 9, 2015
John Stasko
Guest lecturer: Chad Stolper

HW 3

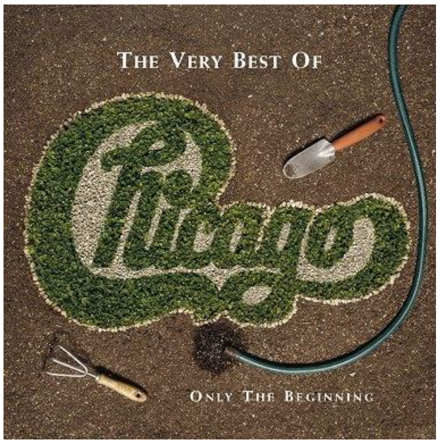
- Three options
  - D3 (tutorial now)
  - Processing (tutorial Friday, when?)
  - Hand-drawn (no tutorial needed)

D3: The Crash Course

D3: The Early Sticking Points

D3: Only the Beginning

D3: Only the Beginning



Please do not be afraid to ask questions!

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

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

9

BUT FIRST….

10

All the stuff you need to know already...

Who has some programming experience?

Who has some web development experience?

Chrome Inspector and Console

- Open the webpage
- Right-click on anything
- Click inspect this element
- Click on the >= button at the top of the inspector to open the console as well
  - (2nd from the left)

15

Starting a Local Webserver

Necessary for Chrome, not for Safari or Firefox

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

16

How many of you have experience with Javascript?

17

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

18

Javascript 101

- All variables are global unless declared using var
  - x = 300 (global) vs. 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( )

19

Javascript 102: Functional Programming

- Javascript is a ***functional language***
  - Functions are themselves objects
  - Functions can be stored as variables
  - Functions can be passed as parameters
- D3 uses these abilities extensively!

20

Javascript 102: Functional Programming

- Javascript is a ***functional language***
  - Functions are themselves objects
  - Functions can be stored as variables
  - **Functions can be passed as parameters**
- D3 uses these abilities extensively!

21

Array.map( )

- Used for applying a function to each element of an array
- The function provided as a parameter takes one parameter itself:
  - d: a/each data point
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map

22

Array.map( )

```
var x = [{pos:1},{pos:2},{pos:3},{pos:4}]
var a = x.map(function(d){
     return d.pos;
})

 a : [1,2,3,4]
```

23

MDN

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

24

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
```

25

SVG BASICS

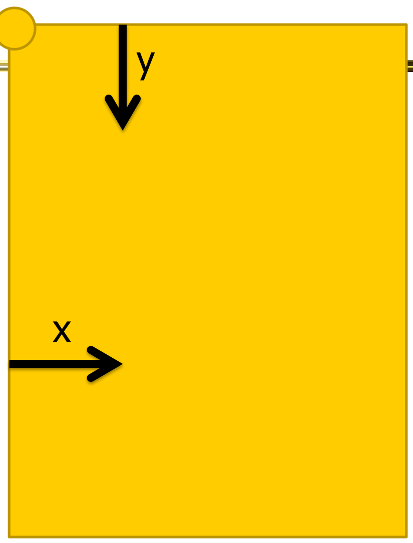
26

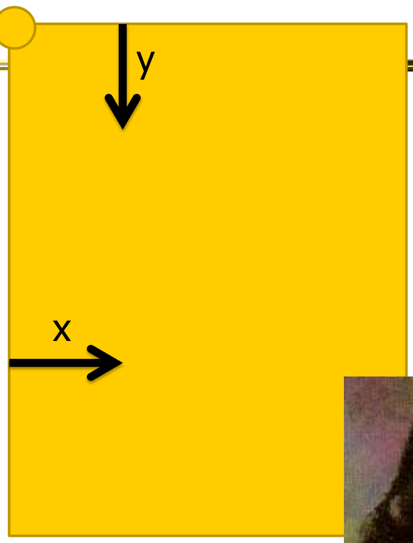
How many of you have experience with SVG?

27

How many have experience with 2D computer graphics (such as Java Swing)?

28





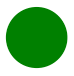
http://smg.photobucket.com/user/Pavan2099/media/RvB/Descart-weeping.png.html

SVG Basics

SVG -> XML Vector Graphics
(Scalable Vector Graphics)

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>

33

SVG Basics

- <svg>
- <circle>
- <rect>

  <path>
- <g>

- <text> (after I've talked about D3)

34

<svg> element

- Overarching canvas
- (optional) Attributes:
  - width
  - height

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

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

35

<svg> element

- Overarching canvas
- (optional) Attributes:
  - width
  - height

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

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

36

<circle> element

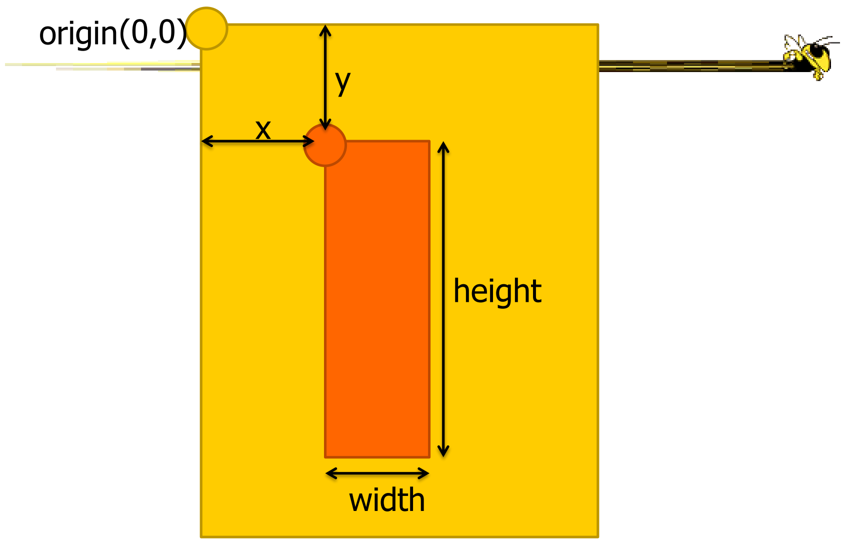
- Attributes:
  - cx (relative to the LEFT of the container)
  - cy (relative to the TOP of the container)
  - r (radius)
- (optional) Attributes:
  - fill (color)
  - stroke (the color of the stroke)
  - stroke-width (the width of the stroke)
- Create with
  - `.append("svg:circle")`

37

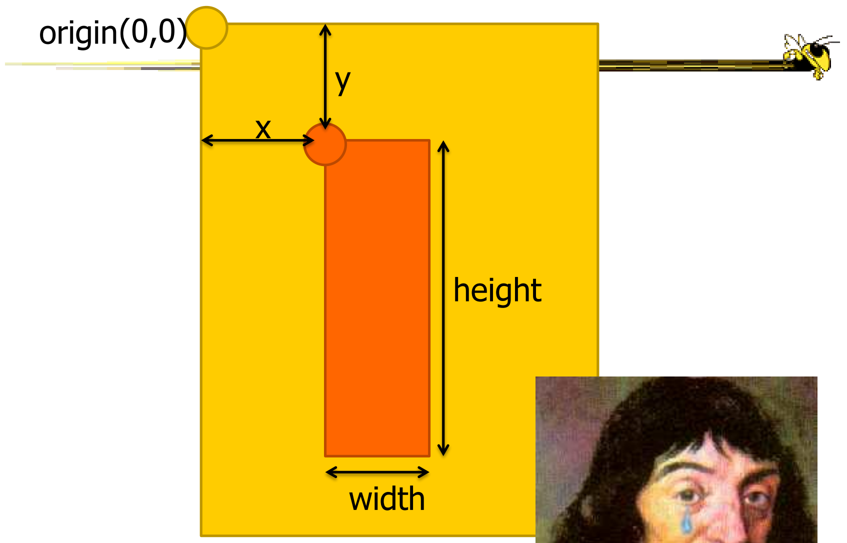
<rect> element

- 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("svg:rect")`

38



39



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

40

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

41

<g> element

- Generic container (Group) element
- Attributes
  - transform
  - (fill,stroke,etc.)
- Create with:
  - `var group = vis.append("svg:g")`
- Add things to the group with:
  - `group.append("svg:circle")`
  - `group.append("svg:rect")`
  - `group.append("svg:text")`

42

CSS Selectors Reference

- #vis → <tag id="vis">
- circle → <circle>
- .canary→ <tag class="canary">
- [color="blue"] → <tag color="blue">

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

43

AND NOW D3…

44

Mike Bostock and Jeff Heer @ Stanford
2009- Protovis

45

Mike Bostock and Jeff Heer @ Stanford
2009- Protovis

46

Mike Bostock and Jeff Heer @ Stanford

2009- Protovis

2011- D3.js

47

Univ. of Washington

Mike Bostock and Jeff Heer @ ~~Stanford~~

2009- Protovis

2011- D3.js

48

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

51

D3

Declarative, domain-specific specification
language for manipulating the DOM

52

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>
```

53

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>
```

54

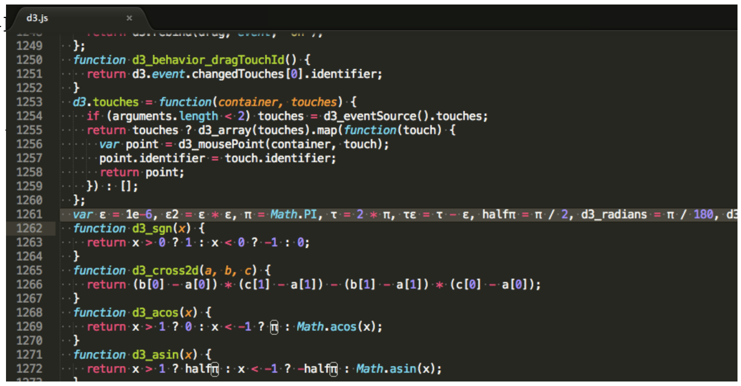
Importing D3

```
<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>
```

55

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) touches = d3_eventSource().touches;
    return touches ? d3_array(touches).map(function(touch) {
      var point = d3_mousePoint(container, touch);
      point.identifier = touch.identifier;
      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);
```

56

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: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
}
```

61

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

62

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

63

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!

64

D3

Declarative, domain-specific specification language for manipulating the DOM

D3

Declarative, domain-specific specification language for manipulating the DOM

Define a template for each type of element

D3

Declarative, domain-specific specification language for manipulating the DOM

Define a template for each type of element
D3 draws one element for each data point

67

Enter-Update-Exit

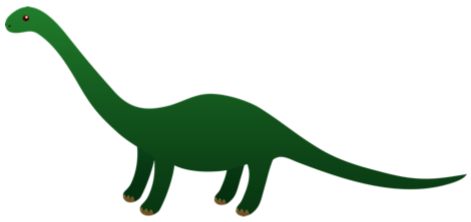
- The *most* critical facet of how D3 works
- If you remember nothing else from today, remember this...
- “Enter-Update-Exit”
- “Enter-Update-Exit”
- “Enter-Update-Exit”

68

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"



69

Enter-Update-Exit

- Pattern:
  - Select a "group" of "elements"
  - Assign data to the group
  - **Enter**: Create new elements for data points that don't have them 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

70

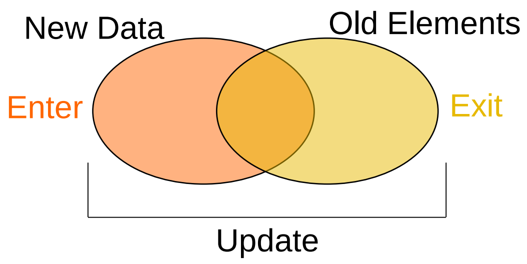
Can be hard to grok:
You can select groups of elements that DON'T EXIST YET

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

71

.enter( ) and .exit( )

- .enter( )
  - New data points
- .exit( )
  - Old elements

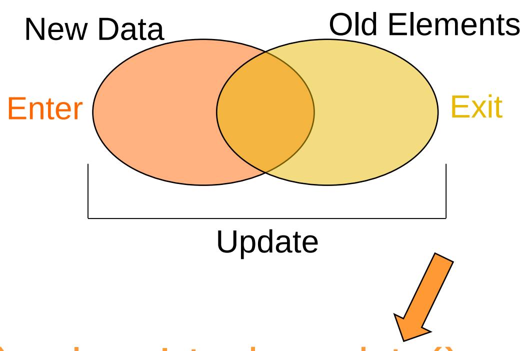


- .enter() and .exit() only exist when .data() has been called

72

.enter( ) and .exit( )

- .enter( )
  - New data points
- .exit( )
  - Old elements

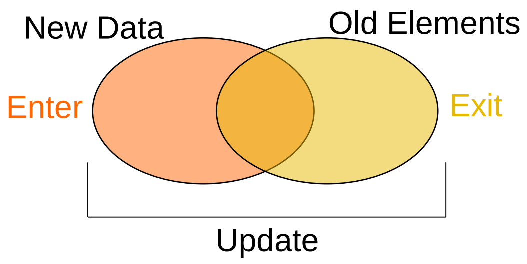


- .enter() and .exit() only exist when .data() has been called

73

.enter( ) and .exit( )

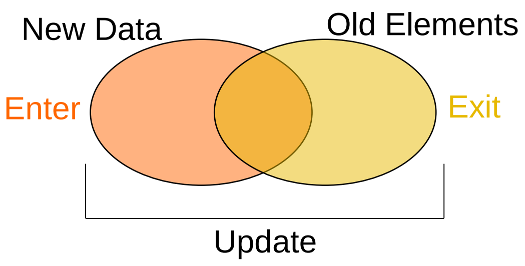
- .data([1,2,3,4])
  - Enter: [1,2,3,4]
  - Update: [1,2,3,4]
  - Exit: [ ]
- .data ([1,2,3,4,5,6])
  - Enter: [5,6]
  - Update: [1,2,3,4,5,6]
  - Exit: [ ]
- .data ([1,2,3])
  - Enter: [ ]
  - Update: ???
  - Exit: [4,5,6]



74

.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]



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("svg:rect") //ENTER!
      .attr( )
      .style( )
group //UPDATE!
      .attr( )
      .style( )
group.exit( ).remove( ) //EXIT!
```

77

WARNING!!!
!

78

E-U-E Pattern Template

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

Many online examples

79

E-U-E Pattern Template

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

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

80

E-U-E Pattern Template

```
var group = vis.selectAll("rect")
      .data(rawdata) //rawdata must be an array!
group.enter( ).append("svg: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!

81

.attr( )

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

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

82

.attr( ) and Functional Programming

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

.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

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

83

<text> elements

84

<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.

85

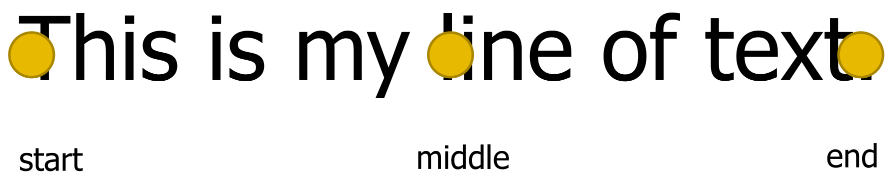
<text> elements

- 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

86

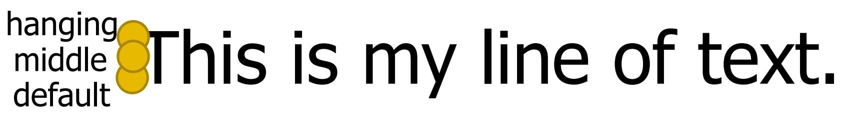
text-anchor style

Where is (0,0)?



dominant-baseline style

Where is (0,0)?



<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")
```

89

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;">
```

90

<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")
```

91

What if you have
two different types of circles?

92

Classing

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

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

- `.attr("height", 5)` is boring
- `.attr("height", function(d,i){ return i*5; })` only works for fixed values
- `.attr("height", function(d){ return d; })` can blow up really quickly…

Scales

95

Scales

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

96

Linear Scales

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

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

y = mx+b

97

Min and Max

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

98

D3

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

99

D3

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

100

Min and Max

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

101

Min and Max

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

- All can be combined with
  - .map(function(d){ }), which returns an [ ]

102

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

103

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

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

104

Linear Scales

- You can even keep the same scale, and just update the domain and/or range as necessary
- Note: This will not ***update*** the graphics all on its own

105

Ordinal Scales

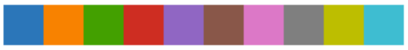
- D3 has built-in color scales!
  - (And they're easy!)
- var colorscale = d3.scale.category10( )
- Also available are:
  - category20( )
  - category20b( )
  - category20c( )
  - (and even a few more)

106

~~Ordinal~~ Categorical Scales

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

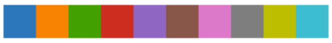
- var colorscale = d3.scale.category10( )



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

~~Ordinal~~ Categorical Scales

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



–

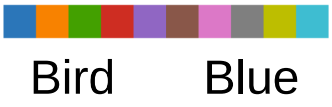
–

–

~~Ordinal~~ Categorical Scales

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

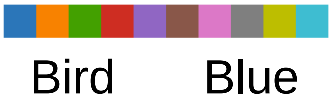
```
    return colorscale(d.type)
 }
```



~~Ordinal~~ Categorical Scales

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

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

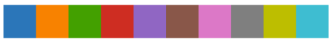


  - `<rect fill=“blue”></rect>`

~~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
Rodent	Orange

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

111

~~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
Rodent	Orange

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

112

~~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)
```

115

D3 even has some
entire techniques built in…
http://bl.ocks.org/mbostock/4063582

116

What if the data is changing?

117

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!
```

118

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!
}
```

119

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!
}
```

120

Transitions

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

121

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
```

122

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.

125

.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.

126

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

129

Good Luck!

chadstolper@gatech.edu

130

Questions?

chadstolper@gatech.edu

Visualization of the Day

- First one up today
- Instructions on website, details on t-square

Project

- Teams set?
- Topic discussions
- Teams & Topics due Monday 14th
  - You must meet me or TA before then
  - Bring 3 copies

HW 2

- Back on Monday

Upcoming

- InfoVis Systems & Toolkits
  - Reading:

    Viegas et al, '07

- Commercial Systems & Demos
  - Reading:

    Spenke & Beilken, '00