D3: The Crash Course

Chad Stolper
Georgia Tech
CSE 6242: Data and Visual Analytics
But first...
Click a name graph to view that name. Double-click to read more about it.
D3: The Crash Course

Chad Stolper
Georgia Tech
CSE 6242: Data and Visual Analytics
D3: Scratching the Surface
D3: Only the Beginning
Please do not be afraid to ask questions!
- History
- Website...
  - Directory Structure
  - Development
- Javascript 101-2
- SVG Basics
- D3.js Crash Course
http://bl.ocks.org/mbostock/1256572
Mike Bostock and Jeff Heer @ Stanford

2009- Protovis
Mike Bostock and Jeff Heer @ Stanford

2009- Protovis

2011- D3.js
Who has some programming experience?
Who has some web development experience?
Website Directory Structure

- (Replace “project” with a real name)

- project/
  - index.html

- project/lib/
  - d3.v3.js

- project/js/
  - project.js

- project/css/

- project/img/
Chrome Inspector and Console

- Open the webpage
- Right-click on anything
- Click inspect this element
- Click on the $\geq$ button at the very bottom to open the console as well
  - (2\textsuperscript{nd} from the left)
Starting a Local Webserver

- `python -m SimpleHTTPServer 8000`
- `http://localhost:8000`
How many of you have experience with Javascript?
Javascript 101

- All variables are global unless declared using `var`
  - `x = 300` (global) vs. `var x = 300` (local)
- Semicolons are completely 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 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!
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!
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

Array.map( )

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

- a : [1,2,3,4]
MDN

- Mozilla Developer Network


- (Easier: google “<command> mdn”)

1/23/14 Chad Stolper

CSE 6242 Guest Lecture
Method Chaining

- Programming paradigm where each method returns the object that it was called on

- Simply put:
  ```javascript
group.attr("x",5).attr("y",5) //returns group
```
  is the same as
  ```javascript
group.attr("x",5) //returns group
group.attr("y",5) //returns group
```
How many of you have experience with SVG?
How many have experience with 2D computer graphics (such as Java Swing)?
SVG Basics

XML Vector Graphics
SVG Basics

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

- W3C Standard
  - [http://www.w3.org/TR/SVG/](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 canvas

- (optional) Attributes:
  - width
  - Height

- Create with
  - d3.select("#vis").append("svg:svg")
<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 width of the stroke)
  - stroke-fill (the color of the stroke)

- Create with
  - .append(“svg: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 width of the stroke)
  - stroke-fill (the color of the stroke)

- Create with
  - `.append("svg:rect")`
Rather than positioning each element, what if we want to position *all* the elements?
<g> element

- Generic container (Group) element

- Attributes
  - transform

- 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”)

Transform Property

“transform”, “translate(x,y)”
`.attr(“transform”, “translate(x,y)”)`
AND NOW D3…
D3

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

- Where to go from here
D3 is a really powerful for-loop with a ton of useful helper functions
D3

- Declarative, domain-specific specification language for visualization

  i.e.
  - Define a template for each type of element
  - D3 draws one element for each data point
Importing D3

<html >
    <head>
        <meta charset="UTF-8">
        <script src='lib/d3.v3.js'></script>
        <script src='js/project.js'></script>
    </head>
    <body>
        <div id="vis"></div>
    </body>
</html>
Assigning the Canvas to a Variable

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

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

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

- `fileloc`: string file location
  - “data/datafile.csv”

- `callback`: function(rawdata){ }
rawdata from a CSV file

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

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

- Ages are Strings, not ints.
- We can fix that:

```javascript
for(var d: rawdata){
    d = rawdata[d]
    d.age = +d.age
}
```
rawdata from a JSON file

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

<table>
<thead>
<tr>
<th>name</th>
<th>school</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam</td>
<td>GT</td>
<td>18</td>
</tr>
<tr>
<td>Barbara</td>
<td>Emory</td>
<td>22</td>
</tr>
<tr>
<td>Calvin</td>
<td>GSU</td>
<td>30</td>
</tr>
</tbody>
</table>
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

- 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”
  - Assign data to the group
  - **Enter**: Create new elements for data points that don’t have them yet
  - **Update**: Set the attributes of all the elements based on the data
  - **Exit**: Remove elements that don’t have data anymore
Can be hard to grok:

You can select groups of elements that DON’T EXIST YET
.enter() and .exit()

- .enter()
  - New data points

- .exit()
  - Old data points
.enter() and .exit()

- .data( [1,2,3,4] )
- .data ( [1,2,3,4,5,6] )
- .data ( [1,2,3] ) //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 function

  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()
  .attr()

//UPDATE!
  .attr()
  .attr()

//EXIT!
  group.exit().remove()
The Attribute Method

Sets attributes such as x, y, width, height, and fill

Technical details:
• rect.attr("x", 5)
• <rect x="5"></rect>
.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>
<text> elements
<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)?

This is my line of text.
dominant-baseline style

Where is (0,0)?

This is my line of text.
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")
  .style("prop1", "val1")
  .style("prop2", "val2")
<ele style="prop1: val1; prop2: val2;">
What if you have two different types of circles?
Classing

- **CSS Classes**
  - Any number of classes per element
  - Select using “.classname”

```javascript
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
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);
})`

- `y = mx+b`
Min and Max

But how do you figure out the min and max for the domain?
A really powerful for-loop with a ton of useful helper functions
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]`

- All can be combined with
  - `.map( function(d){ } ), which returns an [ ]`
d3.min(
    data.map( function(d){ return d.age; })
) // returns the minimum age
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
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 Nominal 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 Nominal 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>`
Axes

D3 also has visual helper-functions
Axes

- `yaxisglyph = chart.append("g")`

```
yaxis = d3.svg.axis()
    .scale(yscale) //must be a numerical scale
    .orient("left") //or "right" or "top" or "bottom"
    .ticks(6) //number of ticks, default is 10
yaxisglyph.call(yaxis)
```
D3 even has some entire techniques built in...

http://bl.ocks.org/mbostock/4062045
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!
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!
}

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)
So they’re no longer static…
But they’re not really interactive…
The on( ) Method
rect.on ("click", function(d) {
    d.color = "blue";
    redraw();
});

HTML Events
- click
- mouseover
- mouseenter
- mouseout
- etc.
Where to get learn more…

- [http://d3js.org/](http://d3js.org/)
  - Tons of examples and basics.


  - List of seemingly ALL the tutorials online

- The Google/StackOverflow combination
  - (my personal favorite)
Thanks!

chadstolper@gatech.edu
Questions?

chadstolper@gatech.edu