

PBO Workshop

Creating Data Visualizations With D3

Ben Racine ¹

¹Cornerstone Systems NW

November 2, 2011

Again, ff you are eager to obtain everything

- Navigate to github.com
- Search for benracine
- This repo should be the first hit, i.e. "d3_cisnet_tutorial"

Again, ff you are eager to obtain everything

- Navigate to github.com
- Search for benracine
- This repo should be the first hit, i.e. "d3_cisnet_tutorial"

Again, ff you are eager to obtain everything

- Navigate to github.com
- Search for benracine
- This repo should be the first hit, i.e. "d3_cisnet_tutorial"

Again, ff you are eager to obtain everything

- Navigate to github.com
- Search for benracine
- This repo should be the first hit, i.e. "d3_cisnet_tutorial"

Attendee Introduction

- Name
- Group
- 2 main visualization tools you have experience with or like
- Any web development experience?
- Any Javascript experience?

Attendee Introduction

- Name
- Group
- 2 main visualization tools you have experience with or like
- Any web development experience?
- Any Javascript experience?

Introductions

Browser Poll

- Chrome
- Firefox 3+
- Safari
- Opera
- IE9
- None of the above?

Introductions

Browser Poll

- Chrome
- Firefox 3+
- Safari
- Opera
- IE9
- None of the above?

Introductions

Browser Poll

- Chrome
- Firefox 3+
- Safari
- Opera
- IE9
- None of the above?

Introductions

Browser Poll

- Chrome
- Firefox 3+
- Safari
- Opera
- IE9
- None of the above?

Introductions

Browser Poll

- Chrome
- Firefox 3+
- Safari
- Opera
- IE9
- None of the above?

Introductions

Browser Poll

- Chrome
- Firefox 3+
- Safari
- Opera
- IE9
- None of the above?

Introductions

Browser Poll

- Chrome
- Firefox 3+
- Safari
- Opera
- IE9
- None of the above?

Javascript in 120 seconds (yeah right)

- C control structures
- Dynamic and weak/duck typing
- Primitive types include:
 - Boolean:
 - `var mayday = false;`
 - Number:
 - `var sal = 20;`
 - `var pal = 12.1;`
 - String:
 - `var myName = "Some Name";`

Javascript in 120 seconds (yeah right)

- C control structures
- Dynamic and weak/duck typing
- Primitive types include:
 - Boolean:
 - `var mayday = false;`
 - Number:
 - `var sal = 20;`
 - `var pal = 12.1;`
 - String:
 - `var myName = "Some Name";`

Javascript in 120 seconds (yeah right)

- C control structures
- Dynamic and weak/duck typing
- Primitive types include:
 - Boolean:
 - `var mayday = false;`
 - Number:
 - `var sal = 20;`
 - `var pal = 12.1;`
 - String:
 - `var myName = "Some Name";`

Javascript in 120 seconds (yeah right)

- C control structures
- Dynamic and weak/duck typing
- Primitive types include:
 - Boolean:
 - `var mayday = false;`
 - Number:
 - `var sal = 20;`
 - `var pal = 12.1;`
 - String:
 - `var myName = "Some Name";`

Javascript in 120 seconds (yeah right)

- C control structures
- Dynamic and weak/duck typing
- Primitive types include:
 - Boolean:
 - `var mayday = false;`
 - Number:
 - `var sal = 20;`
 - `var pal = 12.1;`
 - String:
 - `var myName = "Some Name";`

Javascript in 120 seconds (yeah right)

- C control structures
- Dynamic and weak/duck typing
- Primitive types include:
 - Boolean:
 - `var mayday = false;`
 - Number:
 - `var sal = 20;`
 - `var pal = 12.1;`
 - String:
 - `var myName = "Some Name";`

Javascript in 120 seconds (yeah right)

- C control structures
- Dynamic and weak/duck typing
- Primitive types include:
 - Boolean:
 - `var mayday = false;`
 - Number:
 - `var sal = 20;`
 - `var pal = 12.1;`
 - String:
 - `var myName = "Some Name";`

Javascript in 120 seconds (yeah right)

• Collections

- Array: `var myArray = [[0], [2, 4]];`
- `console.log(myArray[1][1]); > 4`
- Object: `var myObject = {}; myObject.foo = "bar";`

• Functions

- Are objects; have properties and methods
- Can be assigned to variables
- Can be passed as arguments
- Can be returned by other functions
- May be nested
- Closures -> see Python example on wikipedia closure article for a concise example

Javascript in 120 seconds (yeah right)

• Collections

- Array: `var myArray = [[0], [2, 4]];`
- `console.log(myArray[1][1]); > 4`
- Object: `var myObject = {}; myObject.foo = "bar";`

• Functions

- Are objects; have properties and methods
- Can be assigned to variables
- Can be passed as arguments
- Can be returned by other functions
- May be nested
- Closures -> see Python example on wikipedia closure article for a concise example

Javascript in 120 seconds (yeah right)

• Collections

- Array: `var myArray = [[0], [2, 4]];`
- `console.log(myArray[1][1]); > 4`
- Object: `var myObject = {}; myObject.foo = "bar";`

• Functions

- Are objects; have properties and methods
- Can be assigned to variables
- Can be passed as arguments
- Can be returned by other functions
- May be nested
- Closures -> see Python example on wikipedia closure article for a concise example

Javascript in 120 seconds (yeah right)

- Collections

- Array: `var myArray = [[0], [2, 4]];`
- `console.log(myArray[1][1]); > 4`
- Object: `var myObject = {}; myObject.foo = "bar";`

- Functions

- Are objects; have properties and methods
- Can be assigned to variables
- Can be passed as arguments
- Can be returned by other functions
- May be nested
- Closures -> see Python example on wikipedia closure article for a concise example

Really Helpful But Improbable Analogues

"If ProtoVis and jQuery had a Child it Would be d3"

- Any jQuery experience by any chance?
 - d3 is similar, but can also target the SVG (an xml conformant image format)
 - They both do some fancy functional programming to make it possible for us to declaratively reach into the dom tree
- Any Protovis exposure by any chance?

Really Helpful But Improbable Analogues

"If ProtoVis and jQuery had a Child it Would be d3"

- Any jQuery experience by any chance?
 - d3 is similar, but can also target the SVG (an xml conformant image format)
 - They both do some fancy functional programming to make it possible for us to declaratively reach into the dom tree
- Any Protovis exposure by any chance?

Really Helpful But Improbable Analogues

"If ProtoVis and jQuery had a Child it Would be d3"

- Any jQuery experience by any chance?
 - d3 is similar, but can also target the SVG (an xml conformant image format)
 - They both do some fancy functional programming to make it possible for us to declaratively reach into the dom tree
- Any Protovis exposure by any chance?

Really Helpful But Improbable Analogues

"If ProtoVis and jQuery had a Child it Would be d3"

- Any jQuery experience by any chance?
 - d3 is similar, but can also target the SVG (an xml conformant image format)
 - They both do some fancy functional programming to make it possible for us to declaratively reach into the dom tree
- Any Protovis exposure by any chance?

Really Helpful But Improbable Analogues

"If ProtoVis and jQuery had a Child it Would be d3"

- Any jQuery experience by any chance?
 - d3 is similar, but can also target the SVG (an xml conformant image format)
 - They both do some fancy functional programming to make it possible for us to declaratively reach into the dom tree
- Any Protovis exposure by any chance?

Resources

- Github: <http://mbostock.github.com/d3/>
 - API Documentation:
<https://github.com/mbostock/d3/wiki/API-Reference>
 - Examples: <http://mbostock.github.com/d3/ex/>
 - Source: <https://github.com/mbostock/d3.git>
- Google message group
- SVG Specification (v1.1)
- Me *@i3enhamin*

Resources

- Github: <http://mbostock.github.com/d3/>
 - API Documentation:
<https://github.com/mbostock/d3/wiki/API-Reference>
 - Examples: <http://mbostock.github.com/d3/ex/>
 - Source: <https://github.com/mbostock/d3.git>
- Google message group
- SVG Specification (v1.1)
- Me *@i3enhamin*

Resources

- Github: <http://mbostock.github.com/d3/>
 - API Documentation:
<https://github.com/mbostock/d3/wiki/API-Reference>
 - Examples: <http://mbostock.github.com/d3/ex/>
 - Source: <https://github.com/mbostock/d3.git>
- Google message group
- SVG Specification (v1.1)
- Me *@i3enhamin*

Resources

- Github: <http://mbostock.github.com/d3/>
 - API Documentation:
<https://github.com/mbostock/d3/wiki/API-Reference>
 - Examples: <http://mbostock.github.com/d3/ex/>
 - Source: <https://github.com/mbostock/d3.git>
- Google message group
- SVG Specification (v1.1)
- Me *@i3enhamin*

Resources

- Github: <http://mbostock.github.com/d3/>
 - API Documentation:
<https://github.com/mbostock/d3/wiki/API-Reference>
 - Examples: <http://mbostock.github.com/d3/ex/>
 - Source: <https://github.com/mbostock/d3.git>
- Google message group
- SVG Specification (v1.1)
- Me *@i3enhamin*

Resources

- Github: <http://mbostock.github.com/d3/>
 - API Documentation:
<https://github.com/mbostock/d3/wiki/API-Reference>
 - Examples: <http://mbostock.github.com/d3/ex/>
 - Source: <https://github.com/mbostock/d3.git>
- Google message group
- SVG Specification (v1.1)
- Me *@i3enhamin*

Resources

- Github: <http://mbostock.github.com/d3/>
 - API Documentation:
<https://github.com/mbostock/d3/wiki/API-Reference>
 - Examples: <http://mbostock.github.com/d3/ex/>
 - Source: <https://github.com/mbostock/d3.git>
- Google message group
- SVG Specification (v1.1)
- Me *@i3enhamin*

Resources

- Github: <http://mbostock.github.com/d3/>
 - API Documentation:
<https://github.com/mbostock/d3/wiki/API-Reference>
 - Examples: <http://mbostock.github.com/d3/ex/>
 - Source: <https://github.com/mbostock/d3.git>
- Google message group
- SVG Specification (v1.1)
- Me *@i3enhamin*

Participation Query

- Can I get a sense of the fraction that intend on stepping through the exercises?
- Versus the number that would just like to follow along?

Participation Query

- Can I get a sense of the fraction that intend on stepping through the exercises?
- Versus the number that would just like to follow along?

Participation Query

- Can I get a sense of the fraction that intend on stepping through the exercises?
- Versus the number that would just like to follow along?

Clone or Download Slides, Source Code and Tutorials

```
if you have an internet connection
  if you are a git user
    git clone git@github.com:benracine/d3_cisnet_tutorial.git
  else
    https://github.com/benracine/d3_cisnet_tutorial/downloads
  end
else
  we have usb sticks (that also have Chrome on them)
end
```

Fire up your Developer Tools

- In your browser, navigate to exercise-01.html in the tutorials folder of the repo I provided
- Open up your browser's web developer tools
 - Chrome, Safari, Opera and IE9 have built in tools
 - Firebug for Firefox

- ctrl-shift-i in Chrome
- F12 in Firebug/Firefox



href[http://getfirebug.com/wiki/index.php/Keyboard_and_Mouse_S](http://getfirebug.com/wiki/index.php/Keyboard_and_Mouse_Shortcuts)
of What Can Be Done in Your "IDE"

Fire up your Developer Tools

- In your browser, navigate to exercise-01.html in the tutorials folder of the repo I provided
- Open up your browser's web developer tools
 - Chrome, Safari, Opera and IE9 have built in tools
 - Firebug for Firefox
- ctrl-shift-i in Chrome
- F12 in Firebug/Firefox
-

hrefhttp://getfirebug.com/wiki/index.php/Keyboard_and_Mouse_Shortcuts
of What Can Be Done in Your "IDE"

Fire up your Developer Tools

- In your browser, navigate to exercise-01.html in the tutorials folder of the repo I provided
- Open up your browser's web developer tools
 - Chrome, Safari, Opera and IE9 have built in tools
 - Firebug for Firefox

- ctrl-shift-i in Chrome

- F12 in Firebug/Firefox

-

http://getfirebug.com/wiki/index.php/Keyboard_and_Mouse_Shortcuts
of What Can Be Done in Your "IDE"

Fire up your Developer Tools

- In your browser, navigate to exercise-01.html in the tutorials folder of the repo I provided
- Open up your browser's web developer tools
 - Chrome, Safari, Opera and IE9 have built in tools
 - Firebug for Firefox
- ctrl-shift-i in Chrome
- F12 in Firebug/Firefox
-

hrefhttp://getfirebug.com/wiki/index.php/Keyboard_and_Mouse_Shortcuts
of What Can Be Done in Your "IDE"

Canonical Test to Ensure Installation

- Take a peek at:
 - Elements: shows you the current document structure
 - Resources: shows you all documents involved in this rendering
 - Scripts: allows for breakpoints like a classic IDE
 - Console: a REPL to test code
- Enter d3 and you should see *Object* in the response

Canonical Test to Ensure Installation

- Take a peek at:
 - Elements: shows you the current document structure
 - Resources: shows you all documents involved in this rendering
 - Scripts: allows for breakpoints like a classic IDE
 - Console: a REPL to test code
- Enter d3 and you should see *Object* in the response

Canonical Test to Ensure Installation

- Take a peek at:
 - Elements: shows you the current document structure
 - Resources: shows you all documents involved in this rendering
 - Scripts: allows for breakpoints like a classic IDE
 - Console: a REPL to test code
- Enter d3 and you should see *Object* in the response

Canonical Test to Ensure Installation

- Take a peek at:
 - Elements: shows you the current document structure
 - Resources: shows you all documents involved in this rendering
 - Scripts: allows for breakpoints like a classic IDE
 - Console: a REPL to test code
- Enter d3 and you should see *Object* in the response

Canonical Test to Ensure Installation

- Take a peek at:
 - Elements: shows you the current document structure
 - Resources: shows you all documents involved in this rendering
 - Scripts: allows for breakpoints like a classic IDE
 - Console: a REPL to test code
- Enter d3 and you should see *Object* in the response

Canonical Test to Ensure Installation

- Take a peek at:
 - Elements: shows you the current document structure
 - Resources: shows you all documents involved in this rendering
 - Scripts: allows for breakpoints like a classic IDE
 - Console: a REPL to test code
- Enter d3 and you should see *Object* in the response

Briefly playing in the console

- Even if you don't plan on following along, you can do this starter
- Navigate to <http://mbostock.github.com/d3/>
- Let's change the color of the hyperlinks
 - Open console
 - `d3.selectAll("a").style("color","red")`
 - `d3.selectAll("p").style("color","blue")`
 - `d3.selectAll("p").style("color", function(d,i) return "hsl(" + Math.random() * 360 + ",100%,50%)");`
- Note the existence of both `d3.select` and `d3.selectAll`
 - `d3.select` only chooses the first element

Exercise-01.html: Hello World

- This example only uses raw html (i.e. no SVG)
- Note: we're putting html, css and js all in one file for brevity
- Include the main d3 file in line 5
 - This, d3.js, is the 'core' module
 - The default build of d3.js includes:
 - the core
 - scale
 - svg
 - behavior modules
 - Others include:
 - d3.time.js
 - d3.csv.js

Exercise-01.html: Hello World: Selectors

- All d3 commands live in a unified d3 namespace
- A selector, (i.e. `d3.select("body")`), is a key d3 term
 - d3 supports CSS3 selector notation for reaching into the DOM tree
 - Tag ("*div*")
 - Class ("*.awesome*")
 - Identifier ("*#foo*") pause
 - Containment ("*parentchild*")
 - Intersection ("*.this.that*" for logical AND)
 - Union ("*.this, .that*" for logical OR)
 - Attribute ("*[color = red]*")

Exercise-01.html: Hello World: Operators

- Although elements can be selected individually we're normally using operators on the whole set
 - `.text()` is an "operator", another key d3 term
 - Operators can both get or set:
 - attribute: `.attr()`
 - html content: `.html()`
 - CSS classes: `.classed()`
 - a CSS property: `.property()` (Some HTML elements have special properties that are not addressable using standard attributes or styles)

Exercise-01.html: Hello World: Method Chaining

- Notice that method chaining has already begun
- Method chaining takes advantage of functions that are written to return the modified version of the incoming selection
- Elements can be accessed directly
 - (e.g., `selection[0][0]`)
 - or through the `.each()` call

Exercise-01.html: Hello World

- By default, D3 supports svg, xhtml, xlink, xml and xmlns namespaces
- Additional namespaces can be registered
- Operators can be set as either constants or as functions

Bear with me, more examples should solidify this stuff...

Exercise-02.html: Including an SVG Element

- Width and height could be related to the width and height of the window
- Think of the svg element as a canvas with a transformed coordinate system
- A `svg:g` element is means of containing other svg elements

Exercise-02.html: Including an SVG Element: Coordinates

- A transform can be a handy way of moving the coordinate system to a desired location
- Note:
 - Origin is the top-left
 - x is positive to the right
 - y is positive down
 - scales can be used to correct to cartesian coords (more on that to come)

Exercise-02.html: Including an SVG Element: Additional Notes

- `svg:circle` self explanatory
 - Refer to the SVG spec for relevant and/or required circle attributes
- Note the use of a JavaScript namespace variable to cache a selection of interest
- An important design decision
- You want to do this at any crossroads in your workflow

Exercise-03.html: Combining with CSS Selections

- Concepts

- CSS3 selector notation in the style section \approx in the `d3.select("")` command
- Appending is fairly self-explanatory
- Good practice to use intelligent id and class attributes

Exercise-03.html: Combining with CSS Selections

- Namespaces, explain that `svg:svg` ← first one is a namespace, second one is the element itself `svg:g` is kind of like a `div` in `html:...` just a bag in which to group other things in note: you give them uniqueness through class or id
- Attr, addressed in previous slide
- Appropriate use of namespace variables
- Assign a namespace at any "junction" in your workflow i.e. if you're about to add circles AND text to your scenegraph... it's probably appropriate to add a name to the state of your scenegraph at that point

A Quick Break

Exercise-10.html: Bar Chart

- Bar Chart with HTML Elements
- Scales

Exercise-10.html: Bar Chart

- Identity function
- Functional programming
- Data binding selections
- Update
- Enter
- Exit

Exercise-17.html: Dynamic Bar Chart

Probably the last exercise

Exercise-11.html: 2d Array into an HTML Table

- Foo

Exercise-12.html: 2d Array into SVG Bar Chart

- 2d Array into SVG Bar Chart
- RangeBands
- Linear vs. ordinal scales

Exercise-13.html: Axes Elements

- Foo

Exercises-05.html through Exercise-08.html: Skipping for now

- d, i, and this
- Event listeners can take many forms
- Can listen for different types of events
- Click, mouseover, submit, etc.
- There's a subtlety of attaching to multiple functions to the same event.
- i.e. click.foo maps to one function, click.bar maps to another function

Exercises-05.html through Exercise-08.html: Skipping for now

- exercise-05.html: skip tweens and get to data bindings
- exercise-06.html: notice that we're scaling the whole image,
- exercise-07.html: listen to user events, i.e watch the mouse move
- exercise-08.html: mouse fading events
- exercise-09.html: html-based bar-chart to emphasize that it's not just for SVG canvases

Extras

- Transition \approx a non-instantaneous transformation with extra attributes:
 - Duration -
 - Delay -
- Ease
- Interpolate
- Tween (exercise-05.html if we get a chance)
- Call and each for control flow

Conclusion

- You rock for sticking through this duration