Web Apps with Leaflet and D3

Shruti Mukhtyar

@mapchitra

Spatial Data Science Bootcamp

March 25, 2016

Before we start

No VM, we will use Windows for this.

1. Download and extract bootcamp repo from GitHub.

```
https://github.com/berkeley-gif/bootcamp
```

2. Open the command prompt. Navigate to `bootcamp folder`.

```
>cd \Documents\Bootcamp\Day_3\Web_App_with_Leaflet_and_d3

>C:\Python27\ArcGIS10.3\python -m SimpleHTTPServer
```

3. Open Chrome browser. Type in address `localhost:8000` in the URL bar.

Quick Review

- Basic Tools - Code Editor & Chrome Developer Tools
- Quick Review of Client & Server
- HTML
- CSS
- DOM
- JavaScript
- SVG

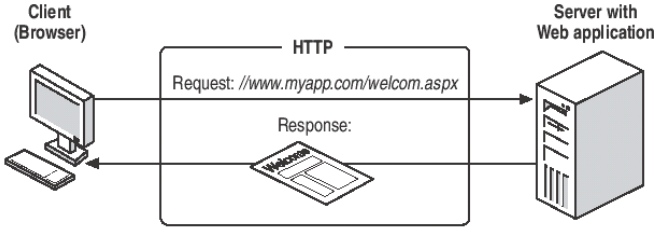
Code Editor

- Sublime Text, Atom, PyCharm, Vim, Notepad++, IDLE, Gedit, TextMate
- Syntax highlighting, indentation, autocomplete, bracket matching
- Run interpreters, debuggers

Browser

- Developer Tools - press `Ctrl+Shift+I` (or `F12`) in Chrome
- Emulator - see how pages look at different breakpoints (screen sizes)
- More info on Chrome Developer tools here

Quick Review of Client & Server



- HTTP, language of the web
- Browser sends HTTP GET Request. Server sends response for each request with content and/or status message.
- Open Network Panel on Chrome Developer Tools. Reload the page to see network acitivity
- How Does the Web Work?

Hyper Text Markup Language (HTML)

- Role » Describes content (not presentation)
- Tags - block and inline

```
<div></div>
<ul>
  <li><a href="gif.berkeley.edu">Home Page</a></li>
  <li><a href="gif.berkeley.edu/people">Staff</a></li>
</ul>
```

- Classes and ID's

```
<div id="map"></div>
<div class="chart"></div>
```

- Properties and Attributes

```
<div id="map" style="height:500px;"></div>
```

- HTML Reference

Cascading Style Sheet (CSS)

- Role » CSS is the presentation of the content
- Selectors, properties, and values
- Properties and Attributes

```
#map {
  height: 500px;
}
h1 {
  font-family: "Helvetica", "sans-serif";
  font-weight: bold;
  background-color: #000;
}
.chart {
  float: left;
}
```

- CSS Reference

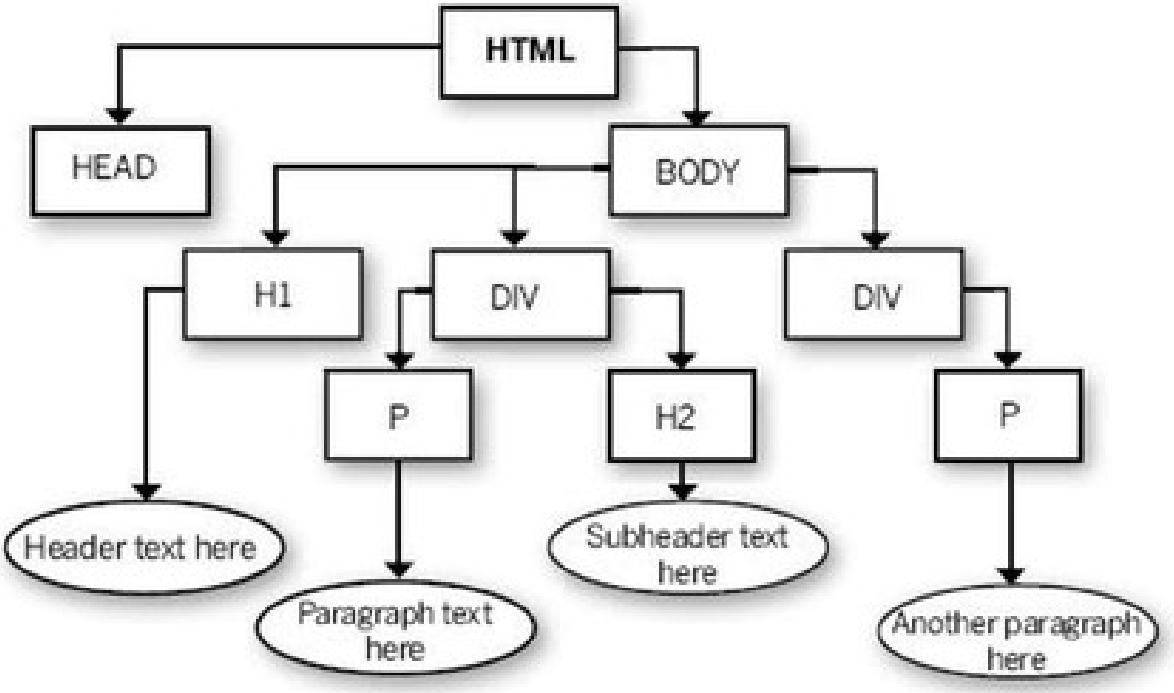
Document Object Model (DOM)

- Structured representation of the document (a tree) created by the browser
- HTML you write is parsed by the browser and turned into the DOM
- Programming interface for HTML and SVG documents
- JavaScript manipulates the DOM
- DOM in simple English, please!

Conceptual web page

```
<!DOCTYPE html>
<html>
<head>
    <!-- head content -->
</head>
<body>
    <h1>Header text here</h1>
    <div>
        <h2>Subheader text here</h2>
        <p>Paragraph text here</p>
    </div>
    <div>
        <p>Another paragraph here</p>
    </div>
    <script>
        <!-- JavaScript content -->
    </script>
</body>
</html>
```

The DOM for conceptual web page



JavaScript

- "Easy to learn, hard to master"
- Role » Creating interaction
- Interpreted by your browser
- Asychronous (code executes in background after client-browser recieves data from server)
- Get familiar with
  - Working with json objects
  - Array functions (forEach, filter, map)
  - Method chaining, Callbacks, Closures, Modules
- JavaScript Reference

Resources

- If you are going to develop medium to large scale web apps learn more about using front-end development tools.
  - Getting Started Web Development Guide
  - Front-end Handbook
- Google Search
  - Favor results from Stack Exchange, Mozilla Developer Network, CSS-Tricks
- caniuse - Check which browsers support what features
- Web Design
  - A List Apart
  - Webdesigner News - curated stories
- Courses for all levels on CodeAcademy, Udacity, Coursera, etc.
- Eloquent Javascript
- Egghead.io - Bite size web dev video training
- Frontend Masters - all levels, paid subscription
- Lot's more on the web!

JSON

```
var obj = {
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": ['Jack','Jill','Bo'],
  "spouse": null
}
```

GeoJSON

```
var obj =
{ "type": "FeatureCollection",
    "features": [
      { "type": "Feature",
        "geometry": {"type": "Point", "coordinates": [102.0, 0.5] },
        "properties": {"prop0": "value0"}
       },
      { "type": "Feature",
         "geometry": {
           "type": "Polygon",
           "coordinates": [
             [ [100.0, 0.0], [101.0, 0.0], [101.0, 1.0],
               [100.0, 1.0], [100.0, 0.0] ]
             ]
         },
         "properties": {
           "prop0": "value0",
           "prop1": {"this": "that"}
           }
      }
    ]
}
```

- GeoJSON Specification, geojson.io, Validate your geojson, Mapshaper

Leaflet

- Lightweight, simple & flexible open source JavaScript mapping library
- Created by Vladimir Agafonkin. Great speaker, checkout some of his talks on Leaflet
- Mobile-friendly. Well documented API, huge amount of plugins.
- Use with other JS mapping libraries (like Esri-Leaflet) or by itself. Similar libraries - OpenLayers, ModestMaps, Polymaps.
- CartoDB.js and Mapbox.js libraries are built on top of Leaflet.
- Leaflet FAQ

What does it do?

- Slippy maps with panning and zooming
- Provides functions for converting data into map layers
- Provides mouse interaction
- Does not provide any data
- You provide tile basemaps and data for overlays
- Easy to extend with plugins

Exercises

- Open command prompt in Windows. Change directory to folder containing exercises. Start a local server using Python module.

```
>cd bootcamp\Day_3\Web_App_with_Leaflet_and_D3\exercises
>C:\Python27\ArcGIS10.3\python -m SimpleHTTPServer
```

- Open Chrome browser. Type `localhost:8000` into the url bar at top. You should see a list of all files in the exercises folder. Follow along on GitHub repo for instructions.

- You will be working in three windows

  - HTML file open in a Chrome browser tab
  - GitHub repo open in a Chrome browser tab
  - Javascript file open in a code editor, e.g. Notepad++

D3

D3 (Data Driven Documents)

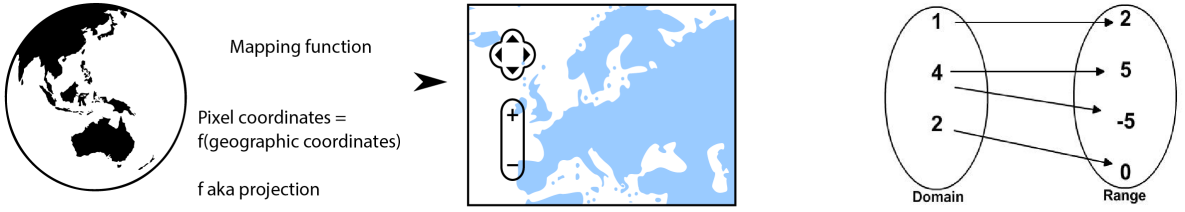
- JavaScript library for producing dynamic, interactive data visualizations in web browsers.
- Makes use of the widely implemented SVG, HTML5, and CSS standards
- Created by Mike Bostock, Vadim Ogievetsky and Jeffrey Heer (all at that time were part of the Stanford Visualization Group).
- Very active user community in Bay Area. Checkout Bay Area d3 User Group.
- Rich ecosystem of examples showing visualization types, coding techniques blockbuilder.org/, bl.ocks.org/
- Navigating the D3 Ecosystem
- How do I learn D3.js? Quora thread

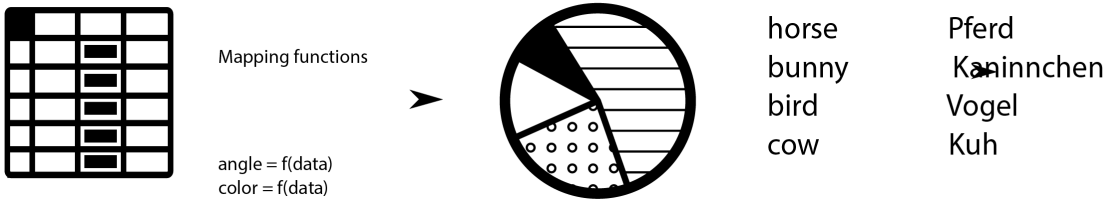
What does it do?

- Transforms data into information
- It is *not* a graphics library, it is *not* a charting library, it is a general purpose data visualization library
- Heavily used in data journalism and visualization. New York Times has been the pioneer in data visualization and data journalism.
- Provides new ways to think about map mapking, communication.

Revisiting mapping

Mapping: An operation that associates each element of a given set (the domain) with one or more elements of a second set (range)





Any (computer) visualization problem can be described as a mapping problem. Geospatial mapping is just a subset.

SVG

```
<svg width="300" height="180">
  <rect x="10" y="20" width="20" height="50" fill="blue"
              stroke="red" stroke-width="1"/>

  <circle cx="100" cy="100" r="25" fill="red"
              stroke="#ddd" stroke-width="5"></circle>

  <g transform="translate(5, 15)">
    <text x="0" y="0">My graphic</text>
  </g>

  <g transform="translate(5, 55)">
    <!-- M: move to (jump)
         L: line to
         Q: curve to (quadratic) -->
    <path d="M0,50 L50,0 Q100,0 100,50"
      fill="none" stroke-width="3" stroke="black" />
  </g>

  <g transform="translate(5, 105)">
    <!-- C: curve to (cubic)
         Z: close shape -->
    <path d="M0,100 C0,0 25,0 125,100 Z" fill="black" />
  </g>
</svg>
```

- Play with this code on SVG Graphic Primitives - JSFiddle

SVG v/s Canvas

- HTML5 Canvas is a raster based format for drawing on the web
  - You can draw raster graphcis with D3
  - You can only get pixel values on click event
  - Faster
- SVG (Scalable Vector Graphics) is a vector based format for drawing on the web
  - HTML has div and span, etc.; SVG has circle and rect, etc.
  - SVG is a DOM for graphical elements
  - You can attach events to SVG elements
  - Most D3 examples work with SVG

Again, what is D3?

- Has everything you need for visualizing complex data BUT you will not find commands like barchart, scatterplot, or piechart, or even map
- Builds vizualizations (and other content) from its basic HTML5 or SVG elements (e.g. < g> , , < r e c t > , < p a th >).
- Most and foremost a DOM manipulation library (in this regard similar to jquery).
- Provides handy utilities for processing data (array, time series, geo data)
- Comes with a lot of functions and methods to create mapping functions that will map your data directly to properties on html elements
- Filled with algorithms (voronoi, quadtrees, circle fitting, convex hull, projections)

"Automating the hard bits you already understand" as opposed to "hiding the hard bits" @andy_matuschak

Data Visualization

- Primary uses
  - Explanation - e.g. Oscar Contenders
  - Exploration - R, Python, Excel, Tableau, D3 (requires some more work)
- Curated lists of data viz and data viz research
  - visualisingdata.com
- Mike Bostock's code blocks
- For geographic visualizations
  - d3.geo
  - Jason Davies, Jason Davies Code Blocks

D3 - Conceptual Hurdles

- Selections
  - How Selections Work - Mike Bostock
- Data binding
  - Pairs and object and an element
  - Keeps track of new and old objects
  - Let's you animate differences between new and old
  - Thinking with joins - Mike Bostock

Next few slides adapted from A fun, difficult intro to d3 - Tom MacWright

D3 Code

- Search result for d3 bar chart
- Let's try to understand this part first (selection & data binding)

```
var svg = d3.select("body").append("svg")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
  .append("g")
    .attr("transform", "translate(" + margin.left + "," + margin.top + ")");

svg.append("g")
  .attr("class", "bars")
  .selectAll("bar")
    .data(data)
  .enter().append("rect")
    .style("fill", "DCDADA")
    .attr("x", function(d) { return xScale(d.date); })
    .attr("width", xScale.rangeBand())
    .attr("y", function(d) { return yScale(d.value); })
    .attr("height", function(d) { return height - yScale(d.value); });
```

D3 Code

- A simpler example

```
d3.selectAll('rect')
  .data([{val: 30}, {val: 55}, {val: 34}])
  .enter()
  .append('rect')
  .attr(Height);
```

```
// an empty selection
// looking for data

d3.selectAll('rect')
```

```
// data, which would be bound to
// a selection
// simple array or array of objects

.data([{value: 30}, {value: 55},
               {value: 34}])
```

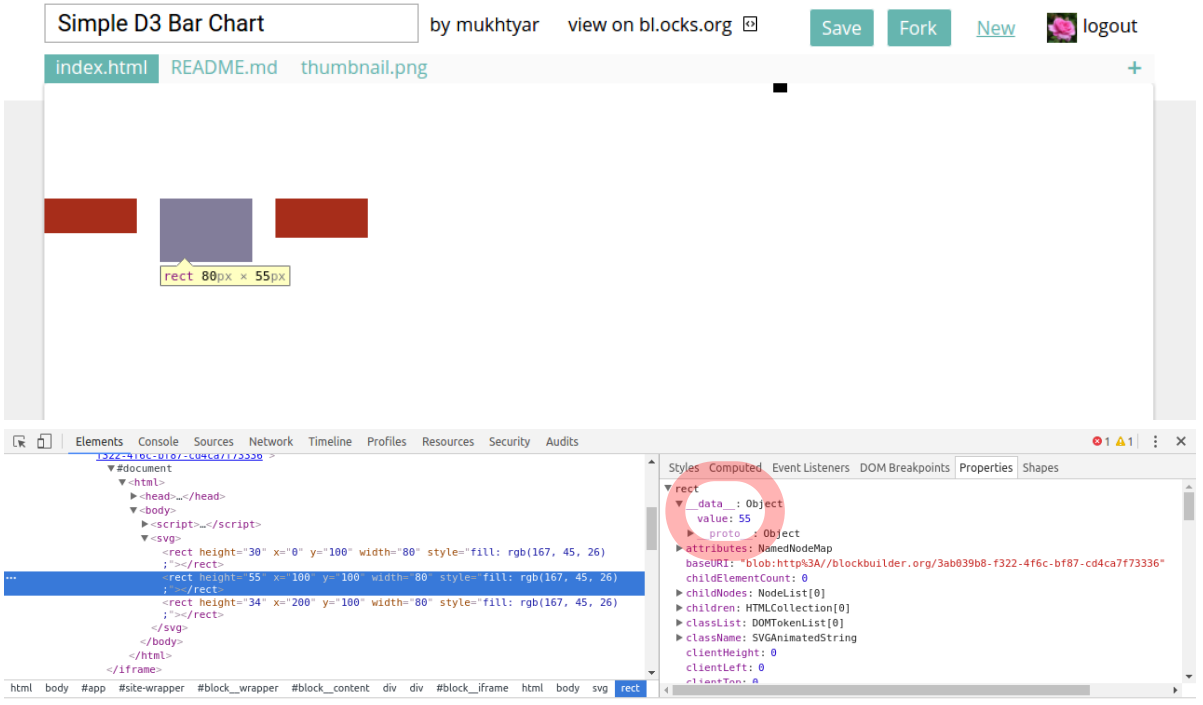
```
// for every time that we see data
// but we do not see an element

.enter()
```

```
// append a SVG element rect
// and set it's height attribute
// to Height
.append('rect')
.attr(Height);
```

What does data binding look like?

- D3 adds a `__data__` property to each graphic element



```
// d3 has a few different
// functions that set stuff
.text()
.property()
.style()
.attr()

// each takes a function
.attr('height', function() { })
```

```
// and that function gets data
// from your .data()
.attr('height', function(d) {
  return d.value;
})

// also provide static values
.attr('height', 100)

// get back existing value
var barHeight = rect.attr('height');
barHeight -> 30/55/34
```

D3 Code

- Same simple example but written slightly differently
- We are now defining a new variable to hold a reference to what d3.selectAll returns
- This makes it easier to work with the subselections (enter, exit, update)

```
// this is an empty selection
// data has been bound to empty selection
var selection = d3.selectAll('rect')
  .data([{value: 30}, {value: 55},
                      {value: 34}]);
```

```
// for each part of the data which
// is not joined to an element in
// the selection,
// add an element
selection
  .enter()
  .append('rect');
```

```
// for each part of the selection
// that is no longer in the data,
// remove it
selection
  .exit()
  .remove();
```

```
// for each part of the selection,
// update some attribute
selection
  .attr('height', function(d) {
    return d.value;
  });
```

Play with this code at Simple D3 Bar Chart - blockbuilder.org

Making it cool - Transitions

- Special type of selection
- Operators apply smoothly over time
- Delays, duration, easing

Play with this code at Simple D3 Bar Chart with Transitions - blockbuilder.org

Working with Transitions - Mike Bostock

Conceptual Hurdles

- Selections
  - How Selections Work - Mike Bostock
- Data binding
  - Thinking with joins - Mike Bostock
- D3 Axes and scales are relatively easier to understand compared to selections and joins

D3 Scale

```
d3.scale.linear()
    // data comes in
    .domain([67, 1098])
    // representation comes out
    //e.g. pixel width/height of svg
    .range([0, 500]);
```

- Use d3 helper methods to calculate domain - d3.extent, d3.max
- Linear, ordinal, categorical and time scales

Using D3 Scale

```
var x = d3.scale.linear()
          .domain([67, 1098])
          .range([0, 500]);

x(0) //-32.49
x(67) //0
x(500) // 209.99
x(1098) // 500
```

- Scale object is also a function
- Calling the scale is how we translate values from one coordinate to another

D3 Axes

```
var x = d3.scale.linear()
          .domain([67, 1098])
          .range([0, 500]);

var xAxis = d3.svg.axis()
  .scale(x)          // x is d3.scale.linear()
  .orient('bottom') // the ticks go below the graph
  .ticks(4);         // specify the number of ticks

var svg = d3.select('body')
  .append('svg')        // create an <svg> element
    .attr('width', 300) // set its dimentions
    .attr('height', 150);

svg.append('g')             // create a <g> element
  .attr('class', 'x axis') // specify classes
  .call(xAxis); // add the axis
```

Using D3 Scale with geo stuff

```
var width = 960,
    height = 500;

var projection = d3.geo.albersUsa()
    .scale(1000)
    .translate([width / 2, height / 2]);

projection([-112, 34])
//[259.40551744936477, 313.5146898982723]
```

- US States Topojson with D3

Using D3 Scale with geo stuff

Building up the mapping function

```
var width = 960,
    height = 500;

var projection = d3.geo.albersUsa()
    .scale(1000)
    .translate([width / 2, height / 2]);

//Creating a path function
var path = d3.geo.path().projection(projection);

// After getting the data
svg.selectAll('.counties')
  .data(counties.features)
  .enter().append('path')
  .attr('d', path)
```

- US Counties Topojson with D3

D3 Layouts

- Pie, Histogram, Stack, Tree, Force, Cluster, etc.
- More at API docs

D3 and other libraries

- Rickshaw, Highcharts, NVD3 are libraries built on top of D3
- D3 and Crossfilter (Fast Multidimensional Filtering for Coordinated Views)
- Open-source tools binding D3 to R, Python
- Vega - higher-level visualization specification language on top of D3

API

- California Health and Human Services Open Data Portal
- Berkeley Ecoengine
- Data.gov
- World Bank
- Weather API's
- PlanetOS
- USGS Earthquakes

Socrata - Company that build data apis for government and non-profits