Mapping Manhattan in D3

Dr Jon Roberts and Dr Jeff Allen

10.16.2013

The Outline for this Evening

- Pick the location you're mapping and get the geoJSON
- Pull in a basic static map
- Tie data to the map and animate
- Zooming - many ways!
- Adding Google maps

Get the code

- Checkout the GitHub repository:
- https://github.com/jonroberts/d3Mapping
- (you can download a zip of the repo, including this talk)

Pick a location

Maps Need Outlines

- D3 needs a geoJSON to get started. Here are some options:

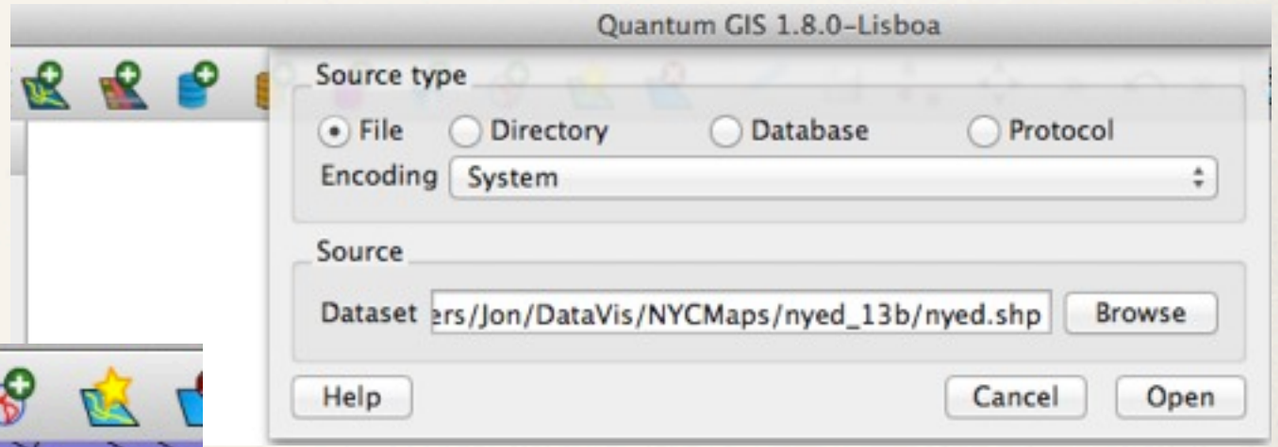
- NYC Neighborhood Outlines

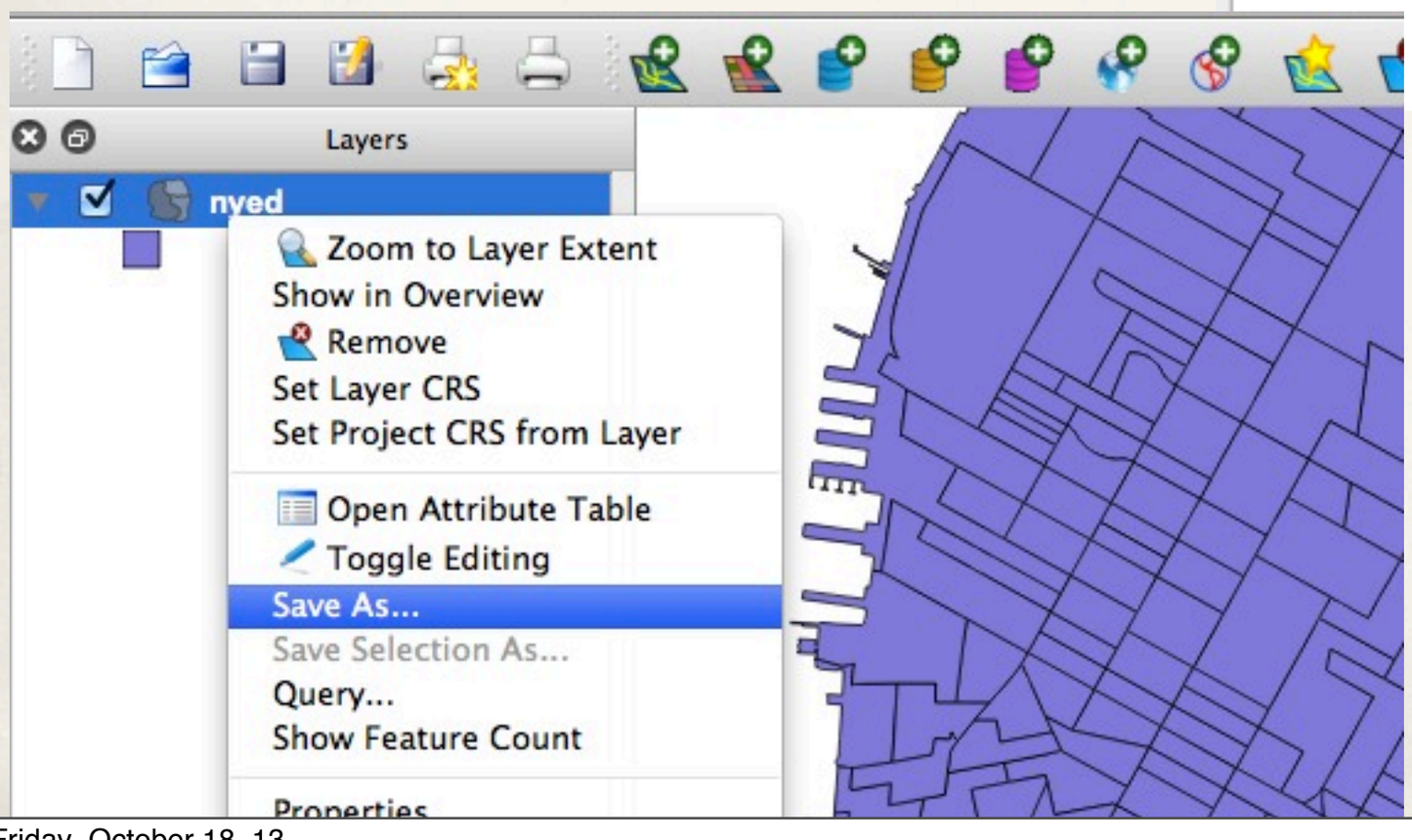
- NYC Zipcodes

- For other outlines - Google, or use QGIS to change Shapefiles to geoJSON

Sideline - QGIS

- If you can only find an ESRI shapefile, QGIS to the rescue:
- Load the shapefile into QGIS
- Right click -> save as... geoJSON

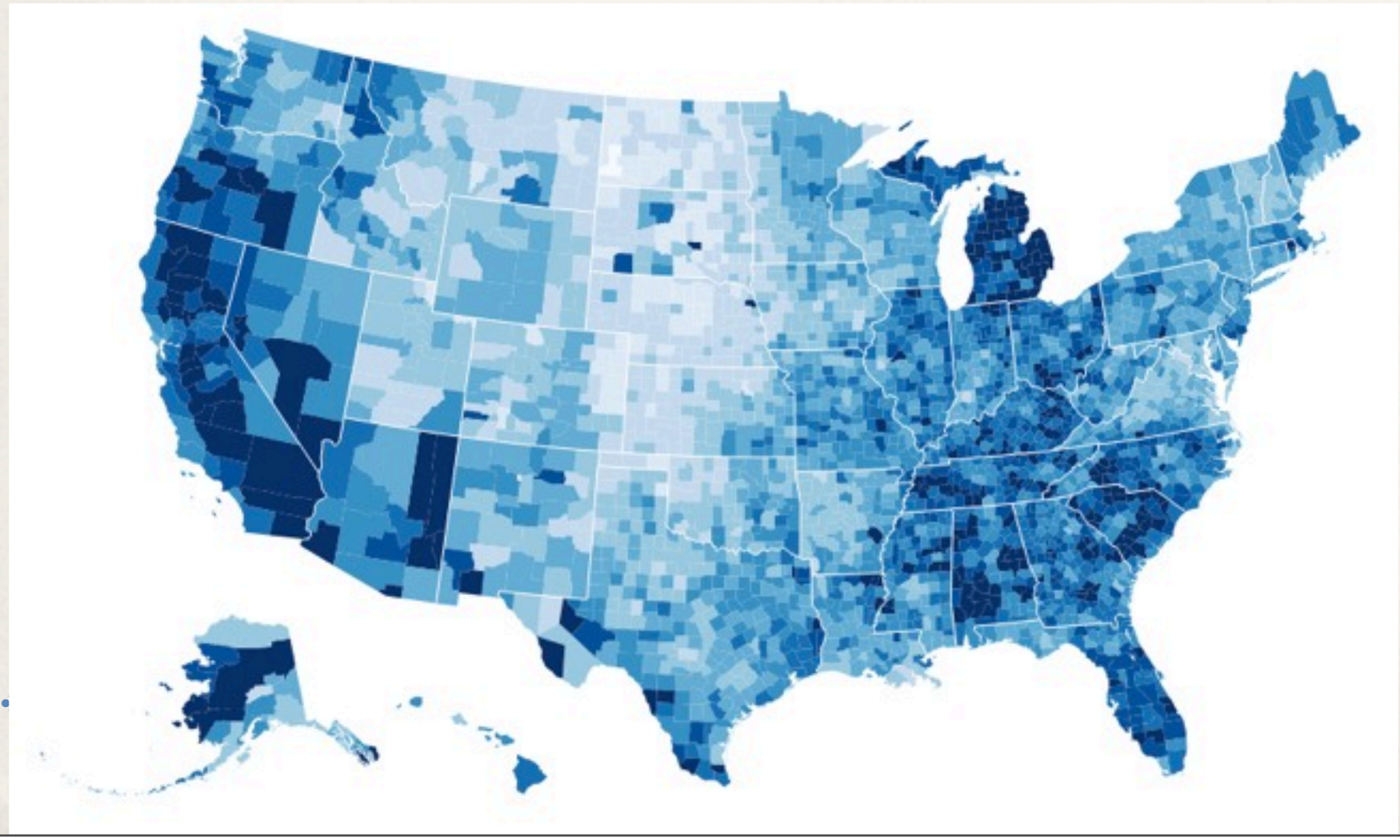




Constructing a basic map

Create a simple map

- Start by ~~stealing~~ borrowing from the D3 examples.
- Let's start with Bostock's Choropleth - see index_1_us.html



Anatomy of a map

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

var rateById = d3.map();

var quantize = d3.scale.quantize()
    .domain([0, .15])
    .range(d3.range(9).map(function(i) { return "q" + i + "-9"; }));

var path = d3.geo.path();

var svg = d3.select("body").append("svg")
    .attr("width", width)
    .attr("height", height);

queue()
    .defer(d3.json, "data/us.json")
    .defer(d3.tsv, "data/unemployment.tsv", function(d) { rateById.set(d.id, +d.rate); })
    .await(ready);

function ready(error, us) {
  svg.append("g")
      .attr("class", "counties")
    .selectAll("path")
      .data(topojson.feature(us, us.objects.counties).features)
    .enter().append("path")
      .attr("class", function(d) { return quantize(rateById.get(d.id)); })
      .attr("d", path);

  svg.append("path")
      .datum(topojson.mesh(us, us.objects.states, function(a, b) { return a !== b; }))
      .attr("class", "states")
      .attr("d", path);
}
```

HTML dimensions of the map

Key / value dictionary to hold TSV data

A shorthand that turns geo outlines into paths

Load data and call ‘ready’ when loaded

Draws the counties

Draws State outlines

Let's go to NYC

Update the Map to NYC

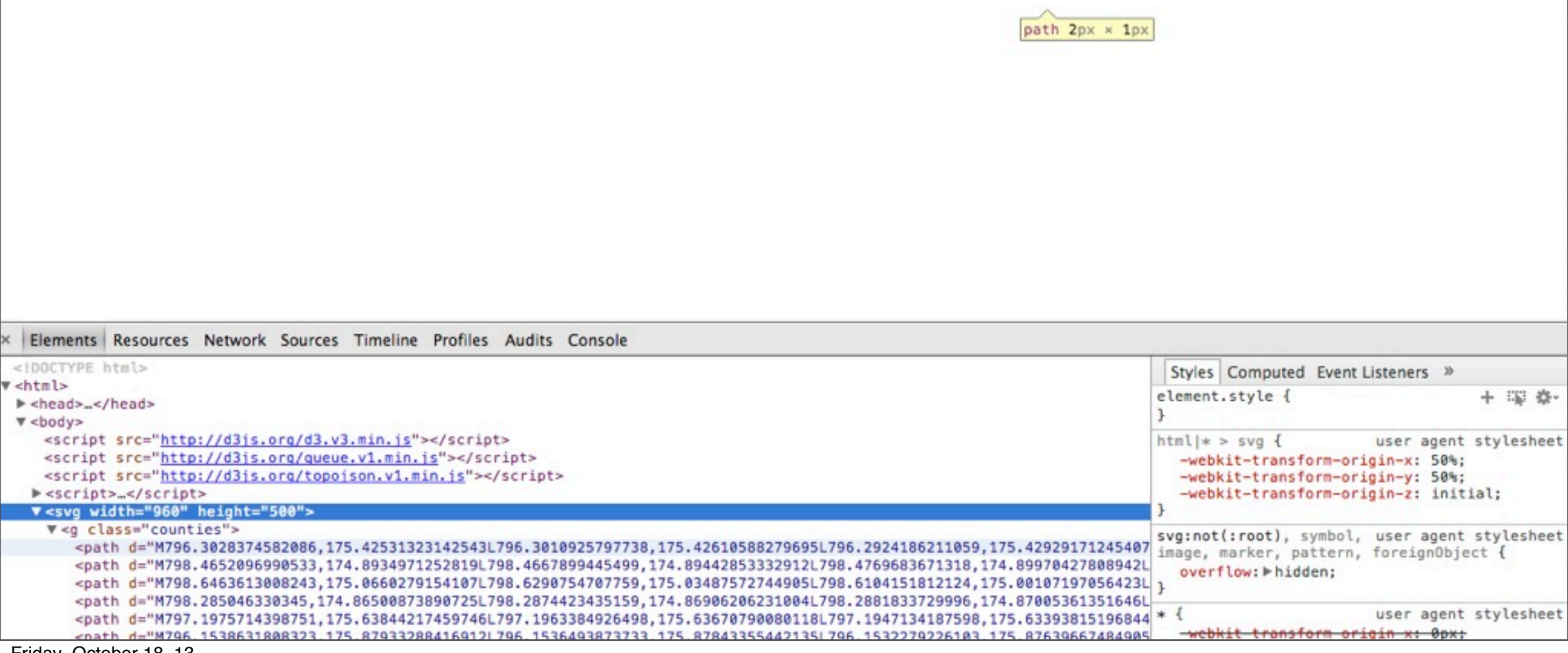
```
var svg = d3.select("body").append("svg")
    .attr("width", width)
    .attr("height", height);

queue()
    .defer(d3.json, "data/nyc-zip-code.json")
    .await(ready);

function ready(error, map) {
  svg.append("g")
     .attr("class", "counties")
    .selectAll("path")
      .data(map.features)
    .enter().append("path")
      .attr("d", path);
}
```

The geoJSON is an array of features. Each feature is a zipcode outline with geometry and properties

```
{
    type: "FeatureCollection",
  - features: [
    - {
          type: "Feature",
        - properties: {
              OBJECTID: 1,
              ZIP: "11372",
              PO_NAME: "Jackson Heights",
              STATE: "NY",
              COUNTY: "Queens",
              ST_FIPS: "36",
              CTY_FIPS: "081",
              BLDGZIP: 0,
              Shape_Leng: 20624.6923165,
              Shape_Area: 20163283.8744
          },
        - geometry: {
              type: "Polygon",
            + coordinates: […]
          }
      },
    - {
          type: "Feature",
        - properties: {
              OBJECTID: 2,
              ZIP: "11004",
              PO_NAME: "Glen Oaks",
              STATE: "NY",
              COUNTY: "Queens",
              ST_FIPS: "36",
              CTY_FIPS: "081",
              BLDGZIP: 0,
```

index_2_nyc.html

But where did the map go?



Re-centering the map

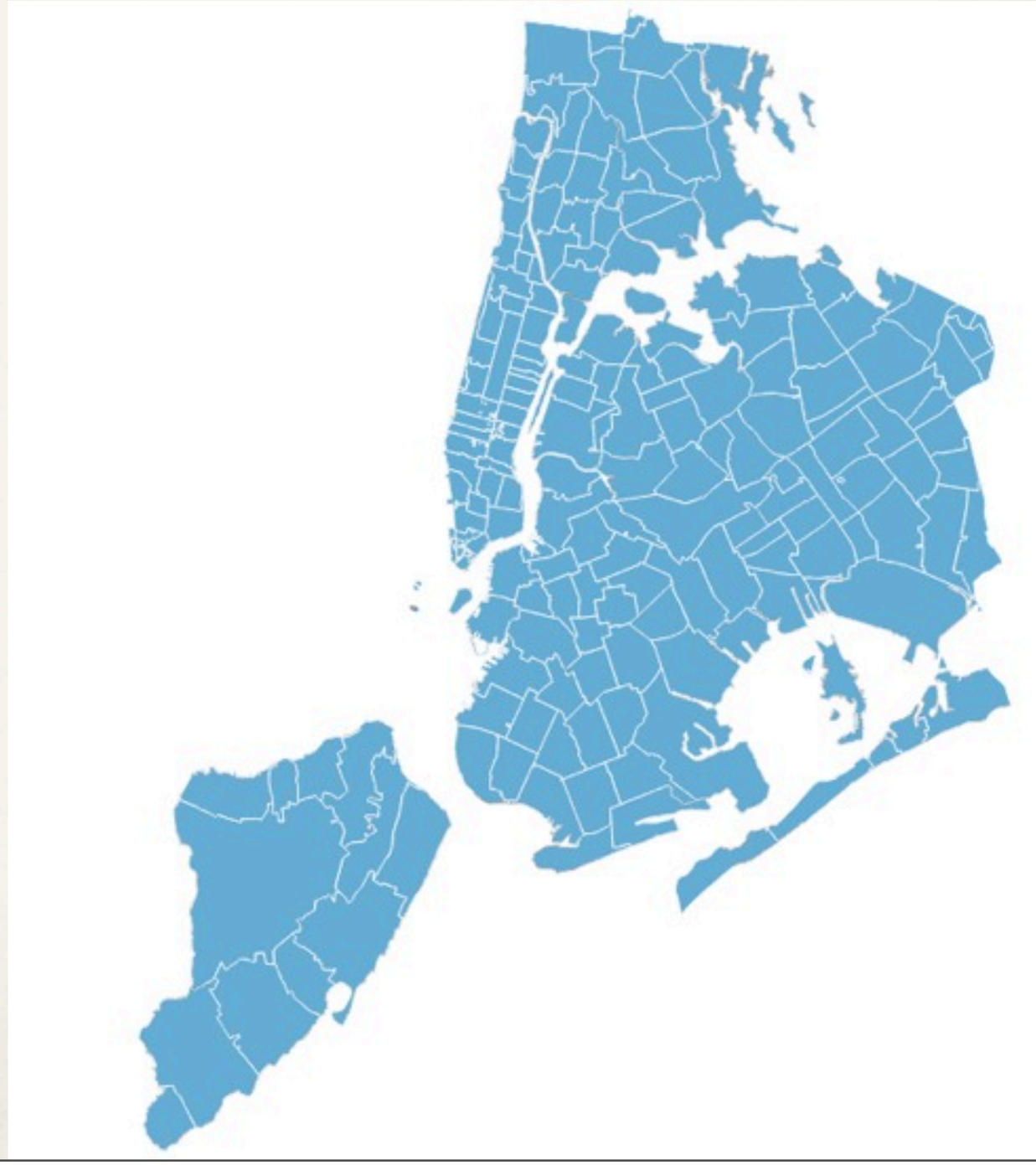
- Need to set up a projection:

```
var projection = d3.geo.albersUsa()
    .scale(82485)
    .translate([-23700,5980]);

var path = d3.geo.path().projection(projection);
```

- Or

```
var projection = d3.geo.conicConformal()
      .parallels([40 + 40 / 60, 41 + 2 / 60])
      .scale(70000)
      .rotate([74, -40 - 45 / 60]);

var path = d3.geo.path().projection(projection);
```

- Use `scale` and `translate` to re-center the map, and hook to the `geo.path` operator.



Tie in Data

Let's hook the color to data

- Read the properties of the JSON to set the color
- Call the function from the `enter()` loop to set the color.
- You can pre-process your geoJSON to populate a data field. Or use `d.properties["ZIP"]` to get the key to pull in data from a Javascript object.

```
function getColor(d){
    // set this color spectrum to be dependent
    // on any parameter of your JSON!
    var ratio=(d["properties"]["Shape_Area"])/1000000.;
    if(ratio>0){
        return d3.hsl(255-ratio,0.4,0.5);
    }else{
        return "lightgrey";
    }
}

function ready(error, map) {
  g=svg.append("g")
      .attr("class", "zipcode")
    .selectAll("path")
      .data(map.features)
    .enter().append("path")
      .attr("d", path)
      .style("fill",function(d){return getColor(d);})
      .on("mouseover",mouseover)
      .on("mouseout",mouseout);

}
```

index_4_nyc_colored.html

Adding a Mouseover Box

- First add event listeners to the zipcode svg paths to trigger on mouse in and mouse out.
- Write functions that show/hide a div on those events
- Fill the div using the data on the zipcode svg

```
function ready(error, map) {
  g=svg.append("g")
      .attr("class", "zipcode")
    .selectAll("path")
      .data(map.features)
    .enter().append("path")
      .attr("d", path)
      .style("fill",function(d){return getColor(d);})
      .on("mouseover",mouseover)
      .on("mouseout",mouseout);
}

function mouseover(d){
    var text="NY"+d.properties["ZIP"];
    // you can add any more information to the mouseover
    // here, using data in your JSON
    $(".mouseover").html(text);
    $(".mouseover").css("display","inline");
}

function mouseout(){
    d3.select("#arcSelection").remove();

    $(".mouseover").text("");
    $(".mouseover").css("display","none");
}
```

Tie div location to mouse location

- Add a listener to the overall html element.
- Get the mouse coords with `d3.mouse(this)`
- Use these coords to set the box location

```
// moves the mouseover box whenever the mouse is moved.
d3.select('html') // Selects the 'html' element
  .on('mousemove', function()
    {
        var locs=d3.mouse(this);    // get the mouse coordinates

        // add some padding
        locs[0]+=15;
        locs[1]+=5;

        $("div.mouseover").css("margin-left",locs[0]);
        $("div.mouseover").css("margin-top",locs[1]);
    });
```

Zooming and Panning

Let's tie zoom to click

- Add a click handler to the zipcode paths
- We have two states - zoomed and centered on the current zip, and not.
- Check and set the zoomed center/translation accordingly.

index_5_nyc_zoom.html

```
  g.selectAll("path")
      .data(map.features)
    .enter().append("path")
      .attr("d", path)
      .style("fill",function(d){return getColor(d);})
      .on("click", click)                    // now we have a click h
      .on("mouseover",mouseover)
      .on("mouseout",mouseout);
}
```

```
function click(d) {
  if (d && centered !== d) {
    var centroid = path.centroid(d);
    x = centroid[0];
    y = centroid[1];
    k = 4;
    centered = d;
  } else {
    x = width / 2;
    y = height / 2;
    k = 1;
    centered = null;
  }
  g.transition()
      .duration(1000)
      .attr("transform", "translate(" + width / 2 + "," + height /
            scale(" + k + ")translate(" + -x + "," + -y + ")")
      .style("stroke-width", 1 / k + "px");
}
```

Panning

- We've set up some globals (I know, but it's only a small app)

```
var x,y,k=1; // parameters to hold the zooming and panning state
var centered;
var sum_dx=0;
var sum_dy=0;
```

- These track zoom level, and the panning.

- Create a d3 behavior and tie it to the svg container:

```
// add a d3 behavior handler.
var drag = d3.behavior.drag()
        .on("drag",function(d){dragging(d);});

var svg = d3.select("body").append("svg")
    .attr("width", width)
    .attr("height", height)
    .call(drag);

function ready(error, map) {
  g=svg.append("g")
      .attr("class", "zipcode")
    .attr("id", "zips")
      .call(drag);
```

Update the map on drag

- Build the total x and y movement
- Transform the svg *container*

```
function dragging(d){
    // first turn off the mouseover
    $(".mouseover").css("display","none");
    // get the deltas that define the relative movement of the cursor
    dx=d3.event.dx/(1.*k);
    dy=d3.event.dy/(1.*k);

    sum_dx+=dx;
    sum_dy+=dy;

    x-=dx;
    y-=dy;

    // update the transformation.
    g.attr("transform", "translate(" + width / 2 + "," + height / 2 + ")\
                         scale(" + k + ")\
                         translate(" + -x + "," + -y + ")")
}
```

http://www.jrsandbox.com/d3Mapping/index_5_nyc_zoom.html

We're Done!

- We have an interactive map
- We can tie new data to arbitrary boundary regions
- We can zoom
- We can pan
- Now go wild! Try new colors, new datasets, new ways of surfacing information.