

Esri Developer Summit

March 8–11, 2016 | Palm Springs, CA



ArcGIS API for JavaScript: Data Visualization

Jeremy Bartley

Jim Herries @jherries

The ArcGIS API for JavaScript lets you build powerful interactive mapping applications. Learn how you can turn your raw data into information with simple layer styling, rich pop-up windows, and interactive data-driven visualizations.

Feature Layer

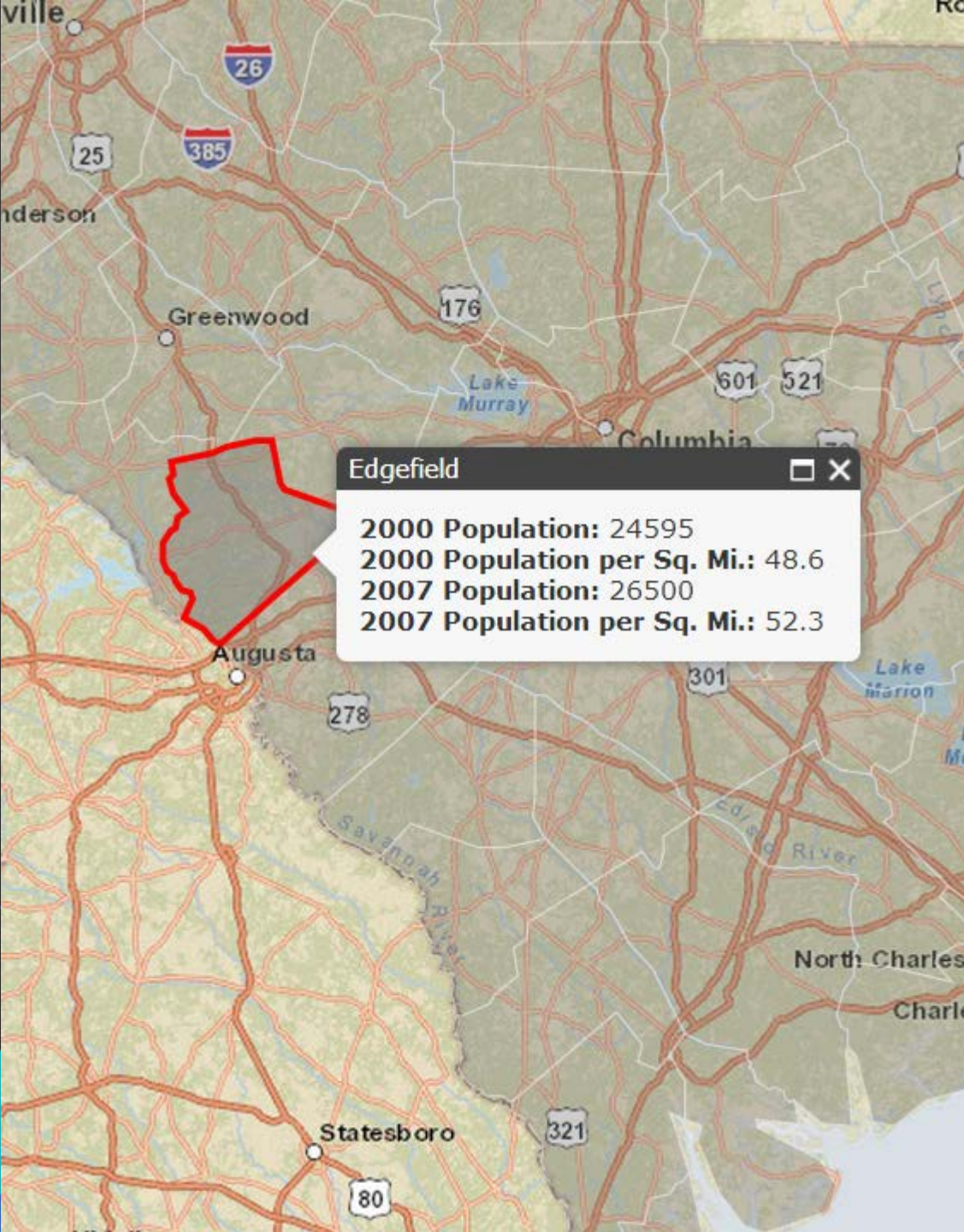
Your gateway to client-side visualization




```
"features" : [  
  {  
    "attributes" : {  
      "ID" : 549,  
      "City" : "Tehama",  
      "Population" : 418,  
      "Household" : 195,  
      "Longitude" : -122.1269008,  
      "Latitude" : 40.0217761,  
      "FID" : 1  
    },  
    "geometry" :  
    {  
      "x" : -13595104.409216635,  
      "y" : 4869107.227782527  
    }  
  },  
  {  
    "attributes" : {  
      "ID" : 12629,  
      "City" : "Lathrop",  
      "Population" : 18023,  
      "Household" : 5261,  
      "Longitude" : -121.3125155,  
      "Latitude" : 37.8152896,  
      "FID" : 2  
    },  
    "geometry" :  
    {  
      "x" : -13504447.452311106,  
      "y" : 4553365.1889916481  
    }  
  }  
]
```

Feature Layer

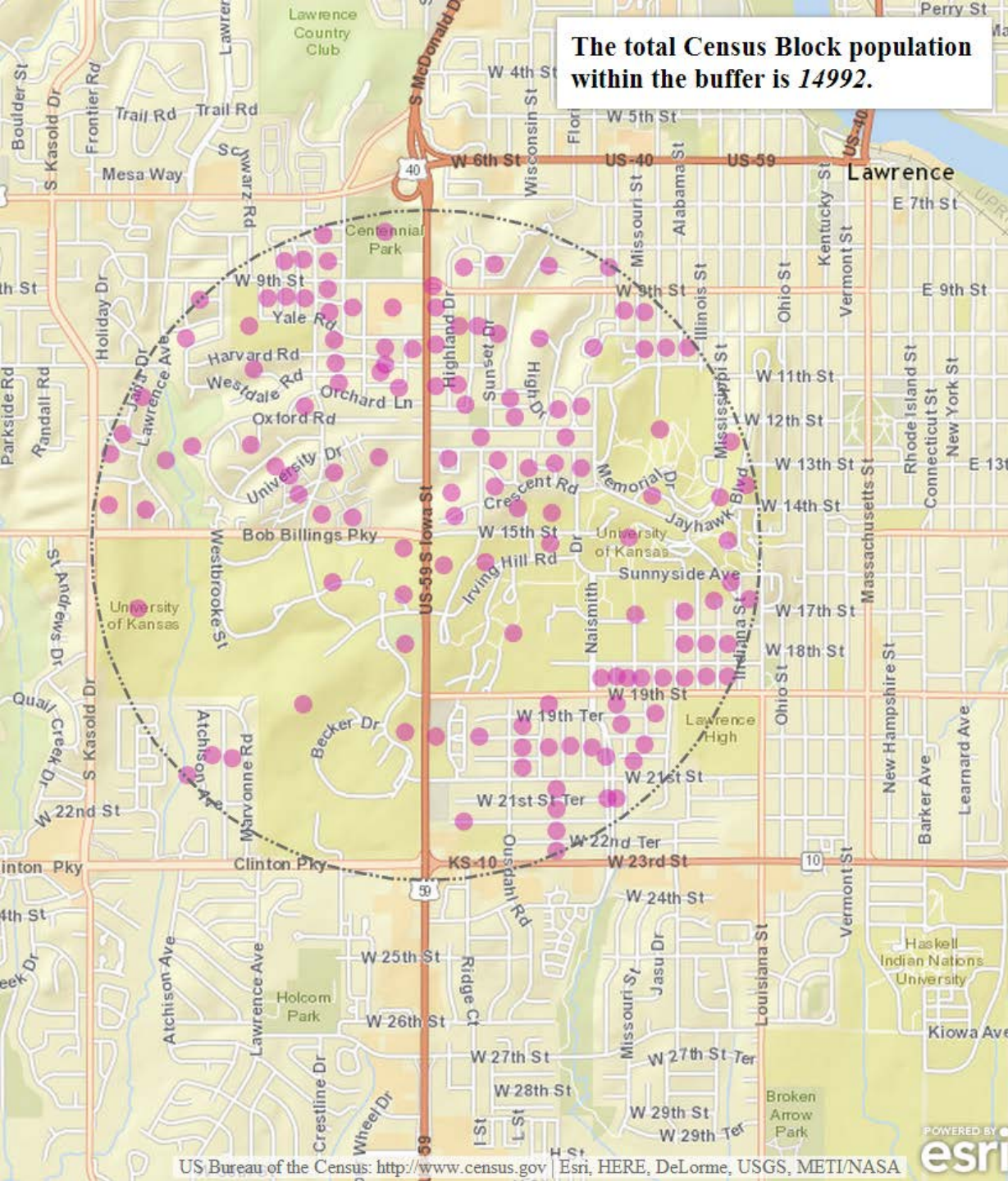
- Vector data (svg or canvas)



Feature Layer

- Vector data (svg or canvas)
- Interactive
- Popups

The total Census Block population within the buffer is 14992.



Feature Layer

- Vector data (svg or canvas)
- Interactive
- Popups
- Query (spatial and attribute)

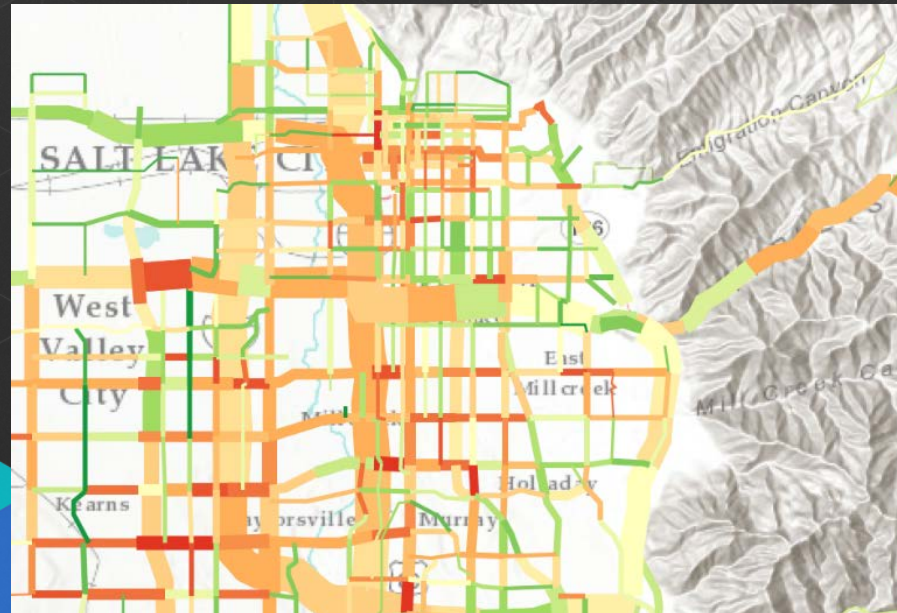
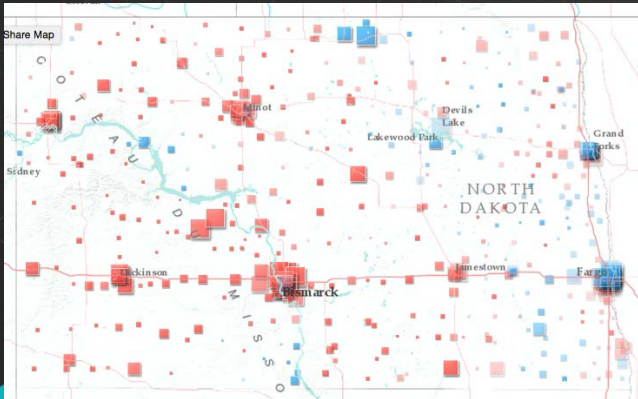
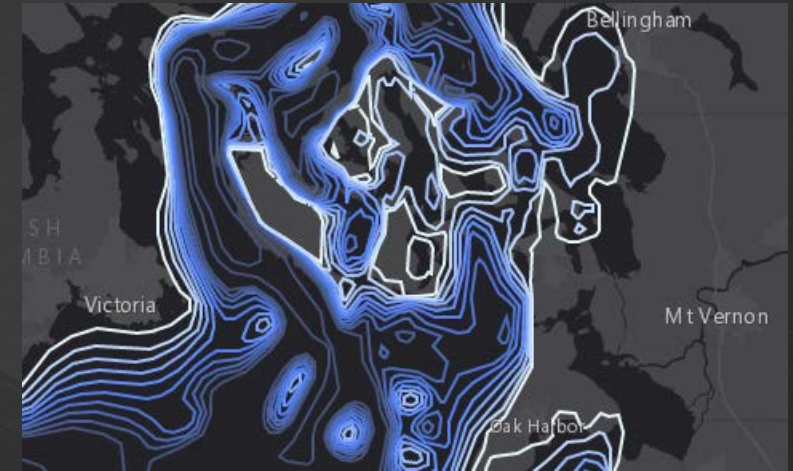
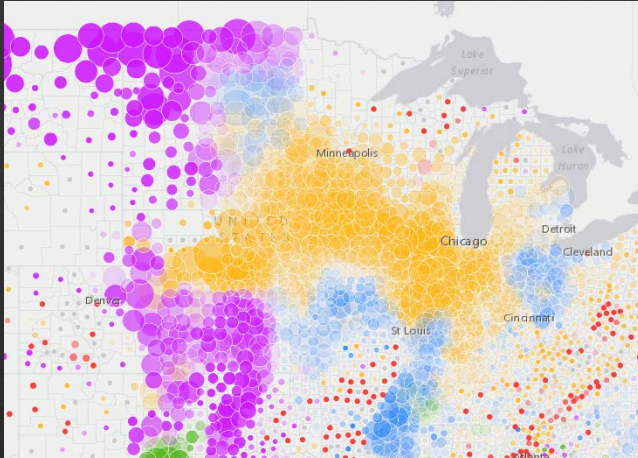




Feature Layer

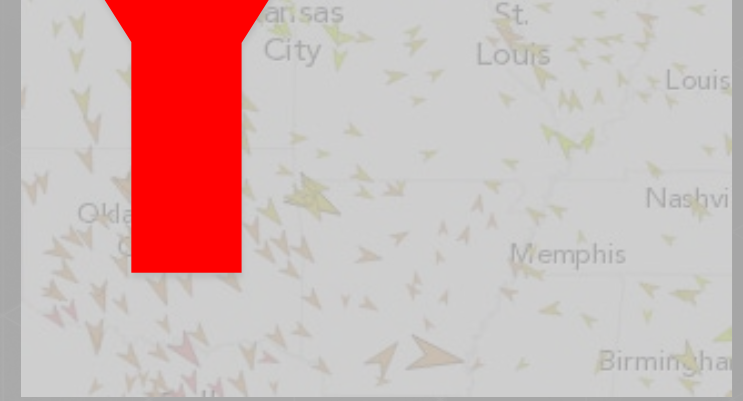
- Vector data (svg or canvas)
- Interactive
- Popups
- Query (spatial and attribute)
- Enables editing

Various Ways to Visualize a Feature Layer



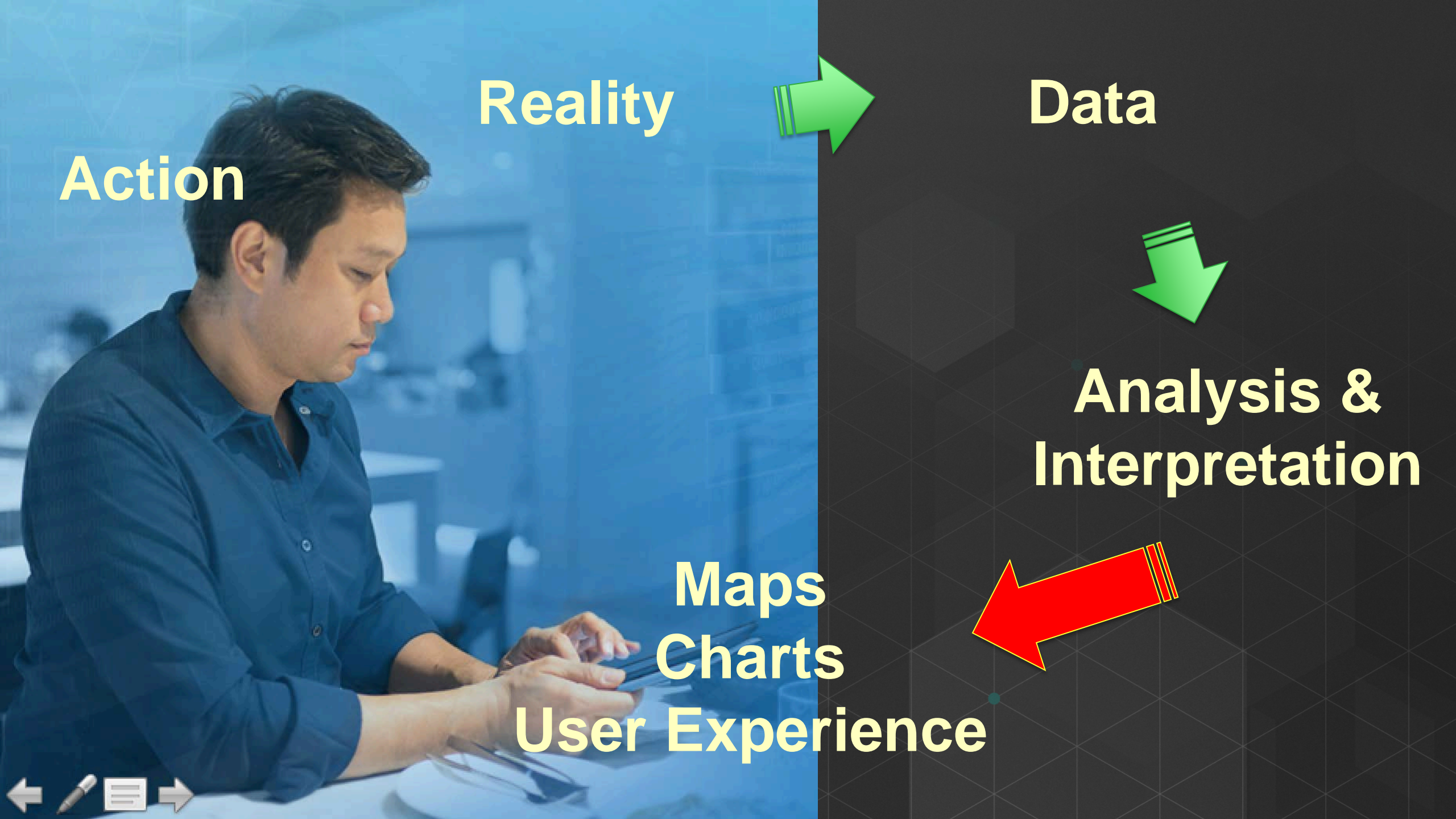
and more!

Various Ways to Visualize a Feature Layer



WHY

and more!



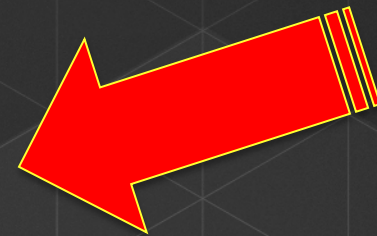
Reality



Data

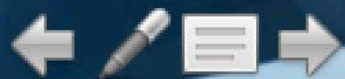


**Analysis &
Interpretation**



Maps
Charts
User Experience

Action





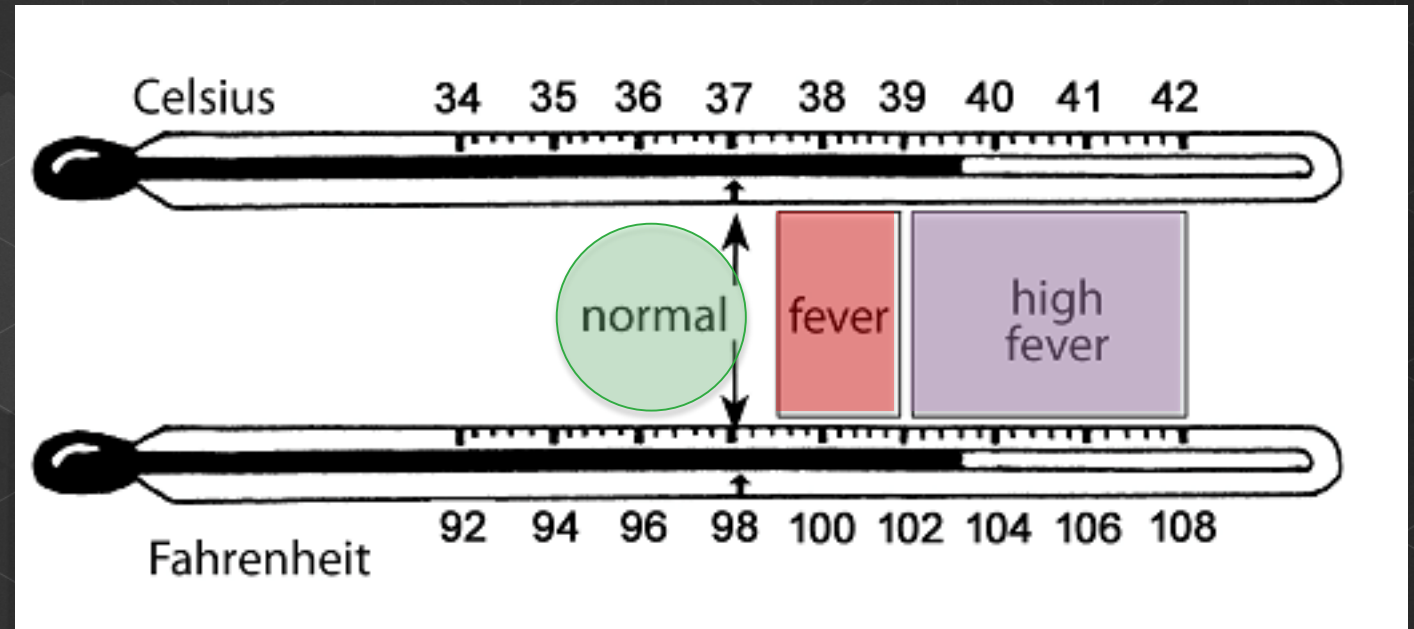
Data

Attendee	Name	BodyTempF	BodyTempC	State
121	Jeremy Bartley	98.6	37.1	normal
122	Jim Herries	98.6	37.1	normal
123	S. Neez	99.1	37.5	fever
124	G.R. Ump-i	103	39.5	high fever

Analysis & Interpretation



User Experience



Renderers...

...reveal anomalies in the data

...help analysis become information

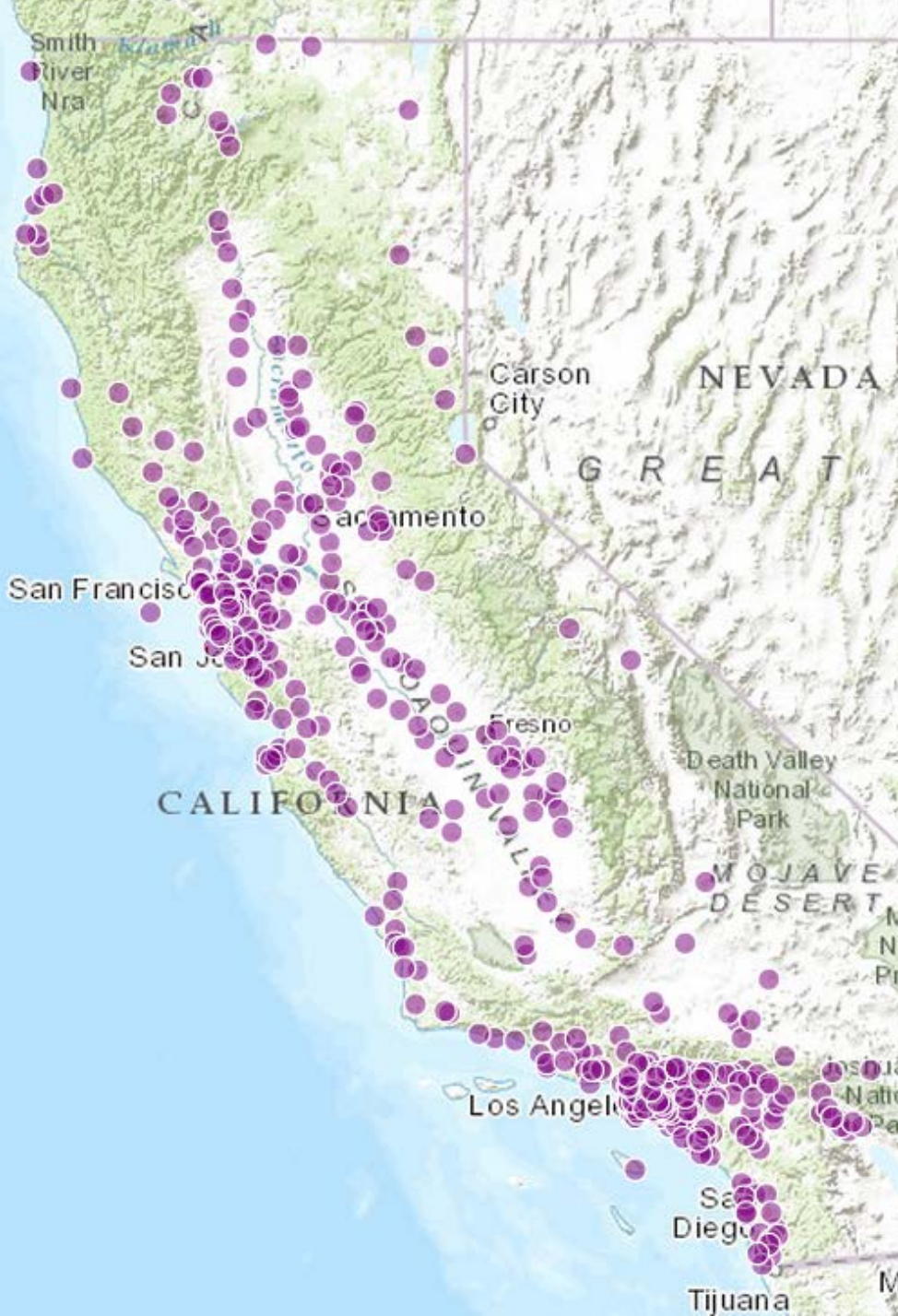
...ensure symbology works **WITH** the data, not against

Renderers

SimpleRenderer

UniqueValueRenderer

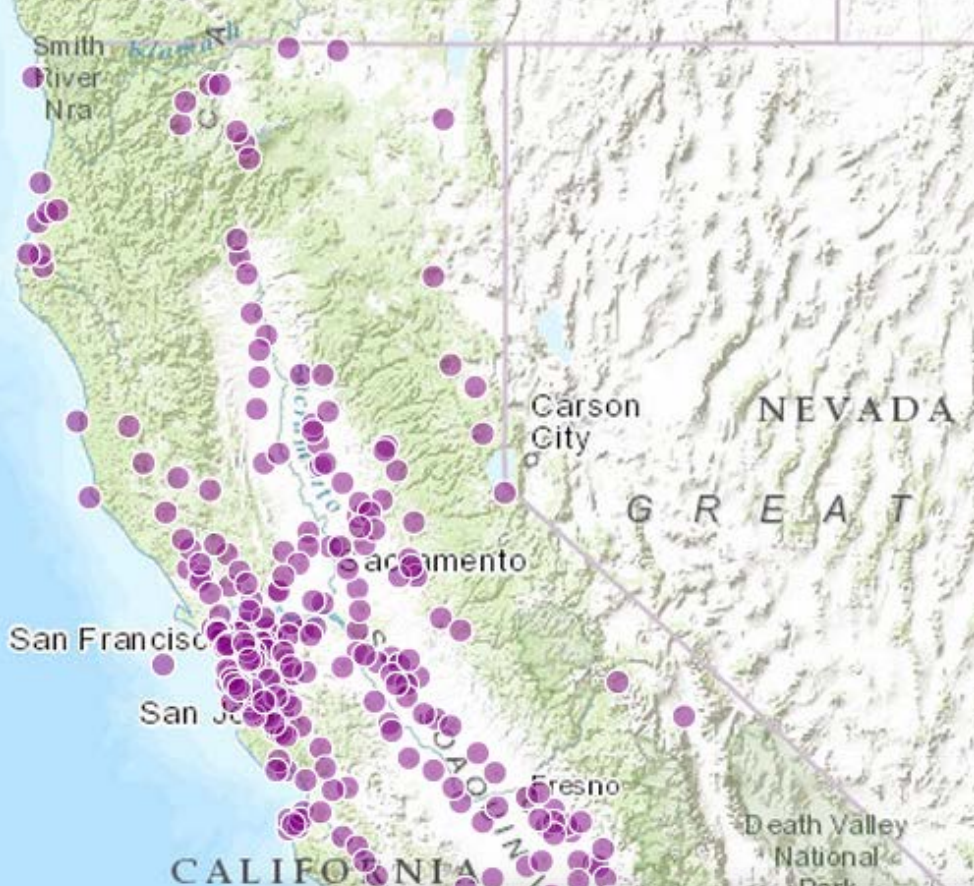
ClassBreaksRenderer



Feature Layer

- A feature layer has its own default symbol
- Defined in the “drawingInfo” property





Feature Layer

- A feature layer has its own default symbol
- Defined in the “drawingInfo” property

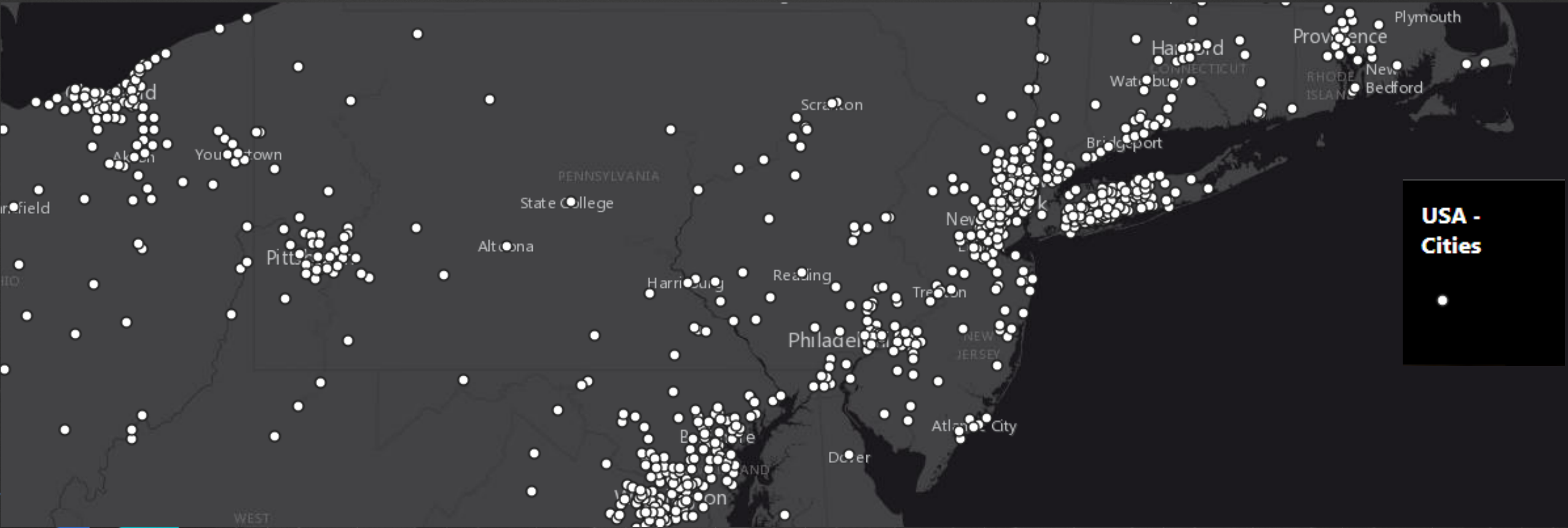
```
var map = new Map("map", {
  basemap: "topo",
  center: [-120, 40],
  zoom: 6
});

var url = "http://services.arcgis.com/V6ZHFr6zdgNZuVG0/ArcGIS/rest/services/CaliforniaCities/FeatureServer/0/";
var layer = new FeatureLayer(url);

map.addLayer(layer);
```


No Field – Show Me Where!

SimpleRenderer



No Field – Show Me Where!

SimpleRenderer



```
var url = "http://sampleserver6.arcgisonline.com/arcgis/rest/services/USA/MapServer/0";
var layer = new FeatureLayer(url);

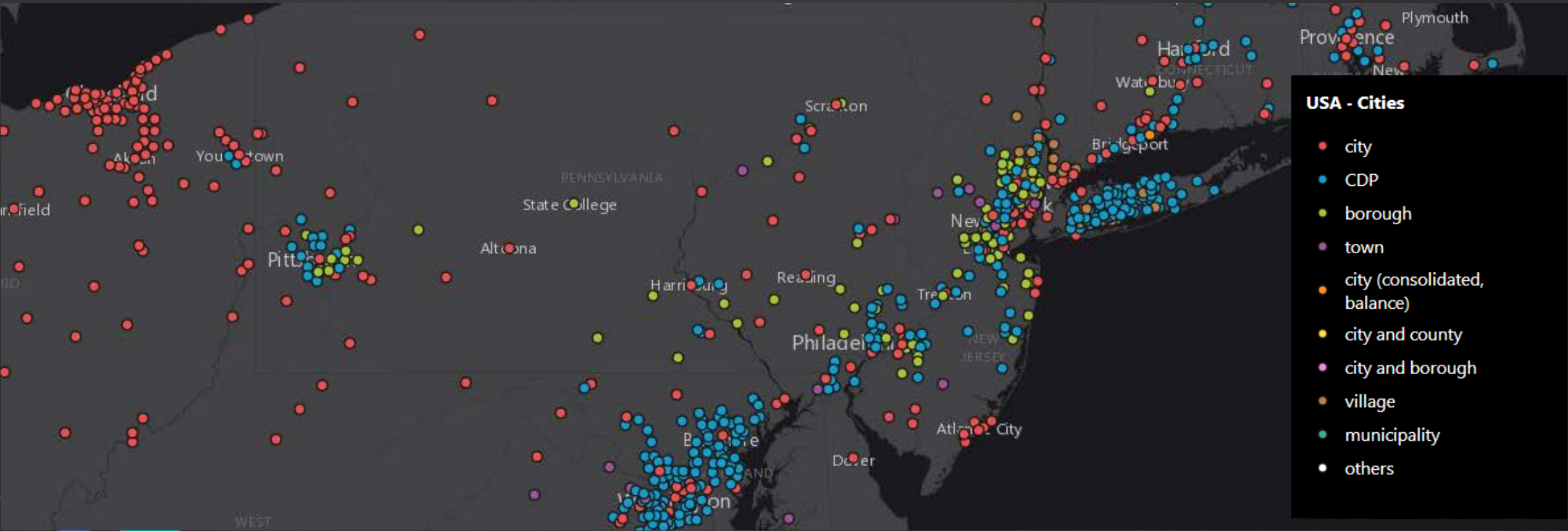
var symbol = new SimpleMarkerSymbol();
symbol.setSize(8);
symbol.setColor(new Color("#fff"));

var outline = new SimpleLineSymbol();
outline.setColor(new Color("#1a1a1a"));
symbol.setOutline(outline);

var renderer = new SimpleRenderer(symbol);
layer.setRenderer(renderer);
map.addLayer(layer);
```


With Field Name – Show Me What!

UniqueValueRenderer



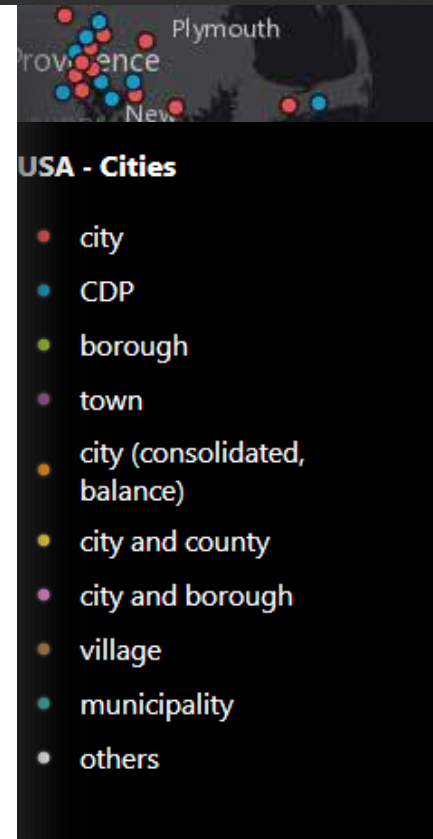
With Field Name – Show Me What!

UniqueValueRenderer

```
var url = "http://sampleserver6.arcgisonline.com/arcgis/rest/services/USA/MapServer/0";
var layer = new FeatureLayer(url);

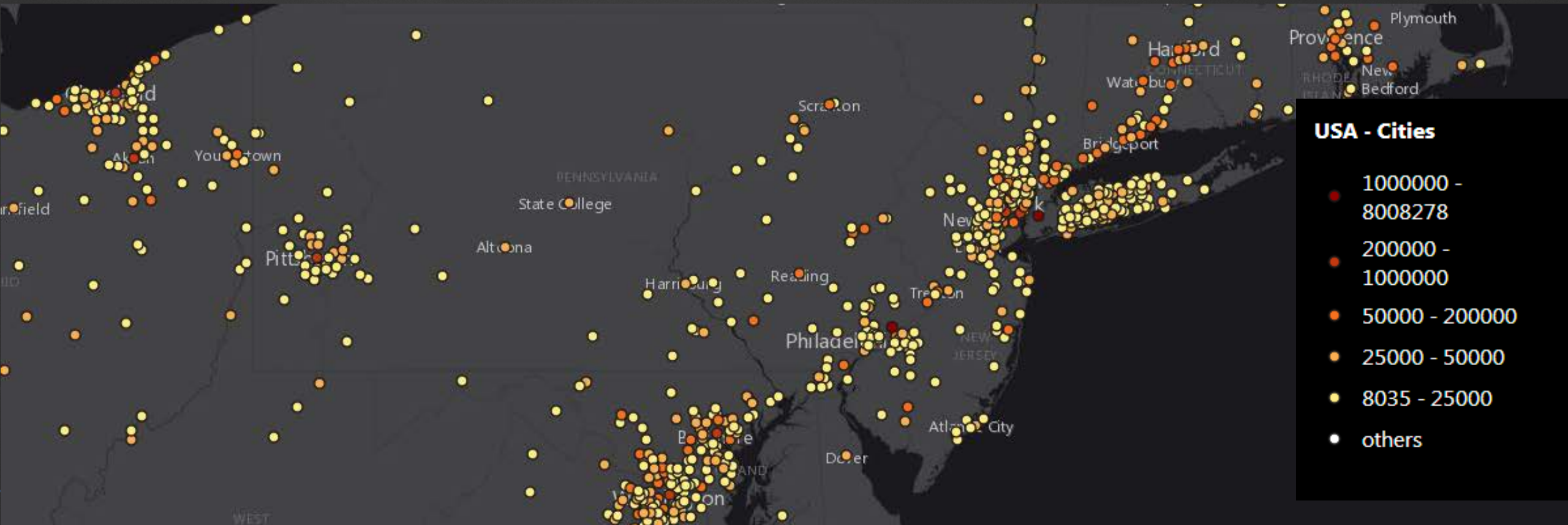
var renderer = new UniqueValueRenderer(createSymbol("#fff"), "class");
renderer.addValue("city", createSymbol("#ed5151"));
renderer.addValue("CDP", createSymbol("#149ece"));
renderer.addValue("borough", createSymbol("#a7c636"));
renderer.addValue("town", createSymbol("#9e559c"));
renderer.addValue("city (consolidated, balance)", createSymbol("#fc921f"));
renderer.addValue("city and county", createSymbol("#ffde3e"));
renderer.addValue("city and borough", createSymbol("#f789d8"));
renderer.addValue("village", createSymbol("#b7814a"));
renderer.addValue("municipality", createSymbol("#3caf99"));
layer.setRenderer(renderer);
map.addLayer(layer);

function createSymbol(color){
  var symbol = new SimpleMarkerSymbol();
  symbol.setSize(8);
  symbol.setColor(new Color(color));
  var outline = new SimpleLineSymbol();
  outline.setColor(new Color("#1a1a1a"));
  symbol.setOutline(outline);
  return symbol;
};
```



With Field Name – Show Me How Much!

ClassBreaksRenderer



With Field Name – Show Me How Much!

ClassBreaksRenderer

```
var url = "http://sampleserver6.arcgisonline.com/arcgis/rest/services/USA/MapServer/0";
var layer = new FeatureLayer(url);

var renderer = new ClassBreaksRenderer(createSymbol("#fff"), "pop2000");
renderer.addBreak(1000000, 8008278, createSymbol("#910000"));
renderer.addBreak(200000, 1000000, createSymbol("#c33910"));
renderer.addBreak(50000, 200000, createSymbol("#f6711f"));
renderer.addBreak(25000, 50000, createSymbol("#fbaf52"));
renderer.addBreak(8035, 25000, createSymbol("#ffed85"));
layer.setRenderer(renderer);
map.addLayer(layer);

function createSymbol(color){
    var symbol = new SimpleMarkerSymbol();
    symbol.setSize(8);
    symbol.setColor(new Color(color));
    var outline = new SimpleLineSymbol();
    outline.setColor(new Color("#1a1a1a"));
    symbol.setOutline(outline);
    return symbol;
};
```



USA - Cities

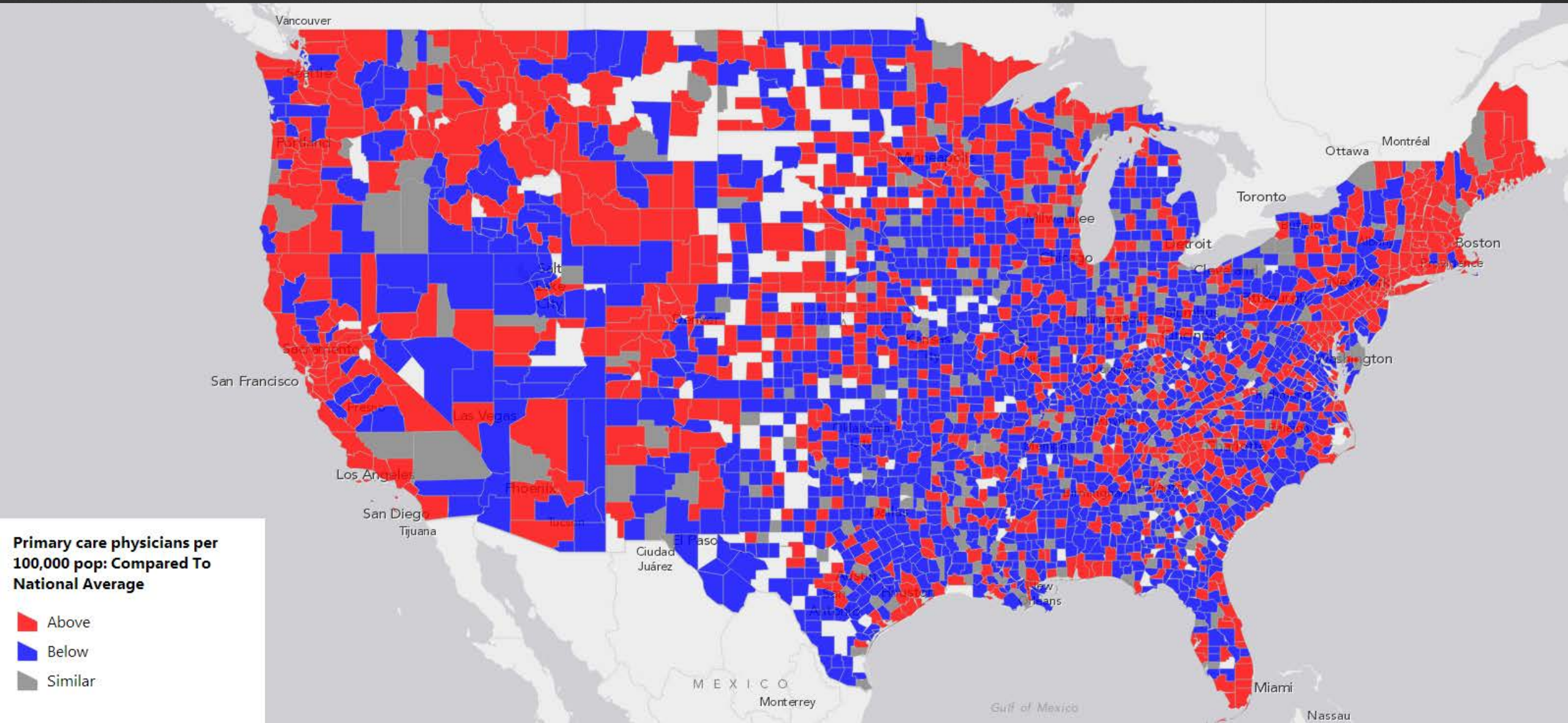
- 1000000 - 8008278
- 200000 - 1000000
- 50000 - 200000
- 25000 - 50000
- 8035 - 25000
- others

With Expression Function

Roll your own

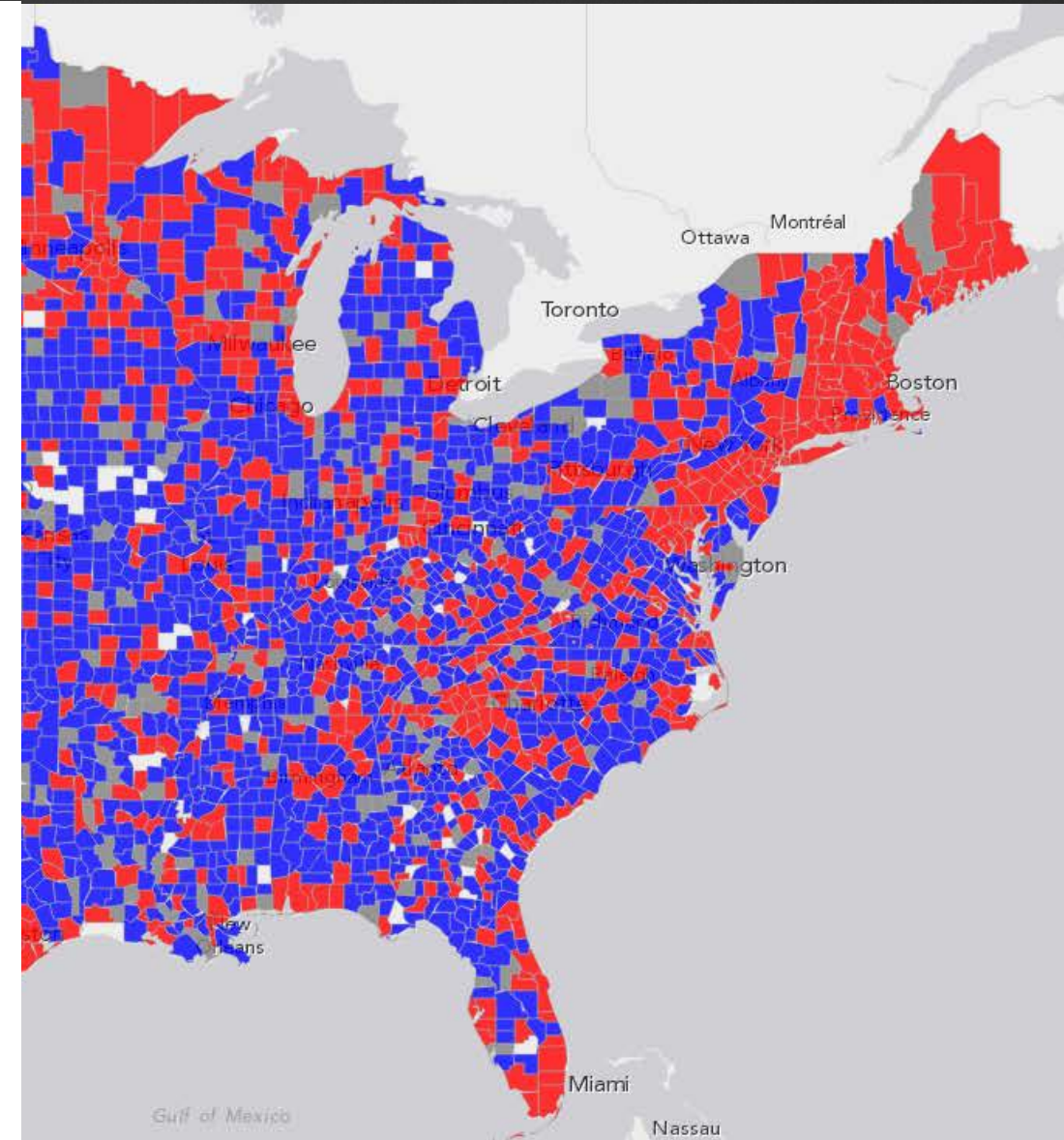
```
var renderer = new UniqueValueRenderer(null, "Prim_Care_Phys_Rate");  
  
var renderer = new UniqueValueRenderer(null, function(graphic){  
    var value = graphic.attributes['Prim_Care_Phys_Rate'];  
    ...  
});
```


With Expression Function – Above and Below



With Expression Function – Above and Below

```
var renderer = new UniqueValueRenderer(null, function(graphic){
  var value = graphic.attributes["SpendingPerPerson"];
  if (value > (avg - fieldStats.halfPercent) && value <= (avg +
    fieldStats.halfPercent)) {
    return "similar";
  } else if (value > (avg + fieldStats.halfPercent)) {
    return "above";
  }
  return "below";
});
renderer.addValue({
  value: "above",
  symbol: new SimpleFillSymbol(SimpleFillSymbol.STYLE_SOLID, outline,
    new Color("red")),
  label: "Above"
});
renderer.addValue({
  value: "below",
  symbol: new SimpleFillSymbol(SimpleFillSymbol.STYLE_SOLID, outline,
    new Color("blue")),
  label: "Below"
});
renderer.addValue({
  value: "similar",
  symbol: new SimpleFillSymbol(SimpleFillSymbol.STYLE_SOLID, outline,
    new Color("gray")),
  label: "Similar"
});
```






With Expression Function – Predominance Mapping

```
var renderer = new UniqueValueRenderer(createSymbol("#d9d9d9"), function(graphic){
  var maxField = "Other";
  var max = 0;
  array.forEach(fields, function(field){
    if (graphic.attributes[field] > max) {
      maxField = field;
      max = graphic.attributes[field];
    }
  });
});
return maxField;
});

renderer.addValue({ value: "M217_07", symbol: createSymbol("#fd7f6f"), label: "Vegetables, acres harvested for sale." });
renderer.addValue({ value: "M188_07", symbol: createSymbol("#b2e061"), label: "Upland cotton, harvested acres." });
renderer.addValue({ value: "M172_07", symbol: createSymbol("#bd7ebe"), label: "All wheat for grain, harvested acres." });
renderer.addValue({ value: "M193_07", symbol: createSymbol("#7eb0d5"), label: "Soybeans for beans, harvested acres." });
renderer.addValue({ value: "M163_07", symbol: createSymbol("#ffb55a"), label: "Corn for grain, harvested acres." });

layer.setRenderer(renderer);
map.addLayer(layer);
```

-  Soybeans for beans, harvested acres.
-  Corn for grain, harvested acres.
-  others

Parametric Properties on Renderer

rotationInfo

sizeInfo

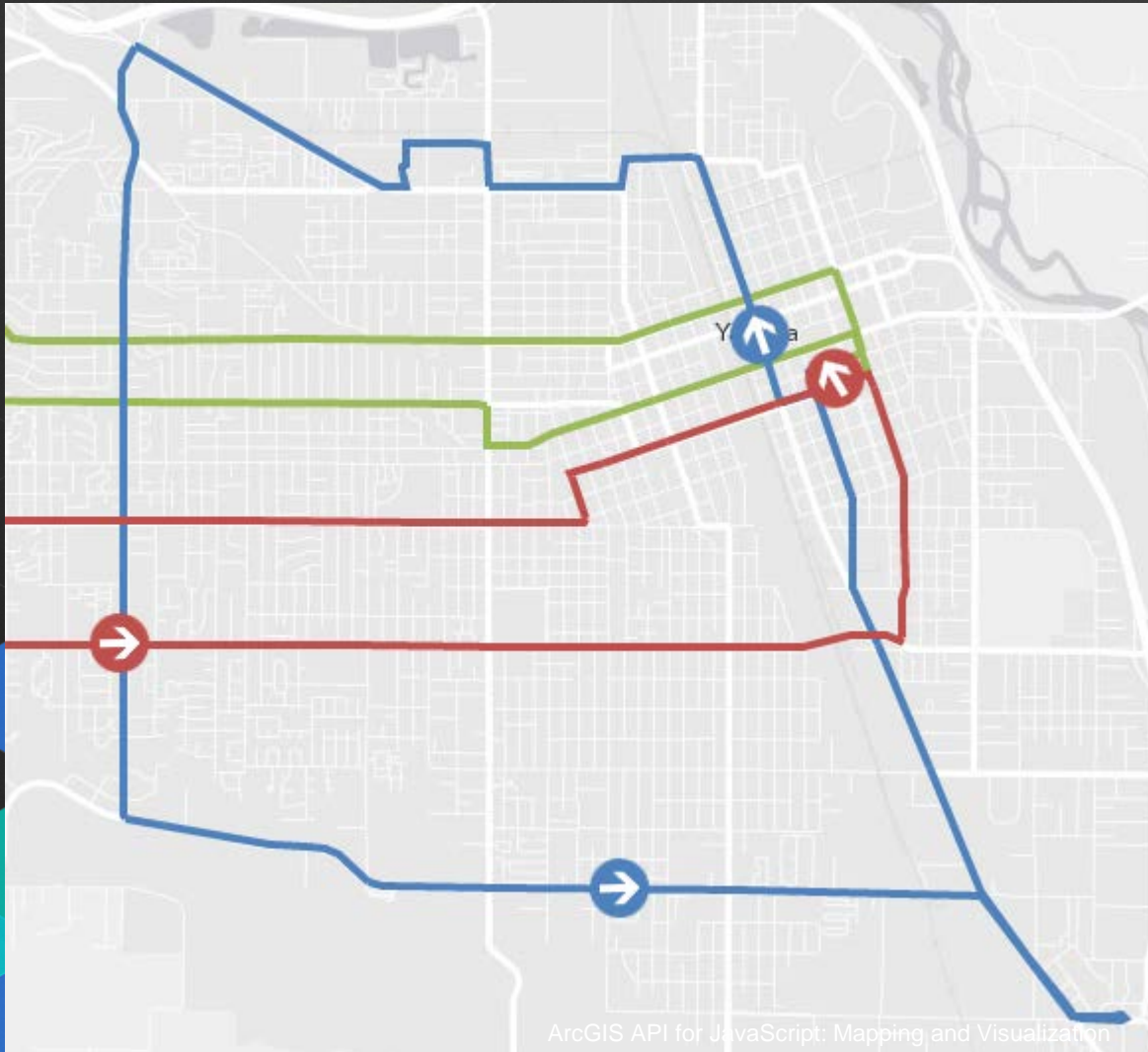
colorInfo

opacityInfo

proportionalSymbolInfo

Rotation

- `Renderer.rotationInfo`



```
layer.renderer.setRotationInfo({  
  field: "heading",  
  type: "geographic"  
});
```



Geographic



Arithmetic

Varying Size – By Real-World Distance/Area

- SizeInfo visual variable

```
var renderer = new SimpleRenderer();
```

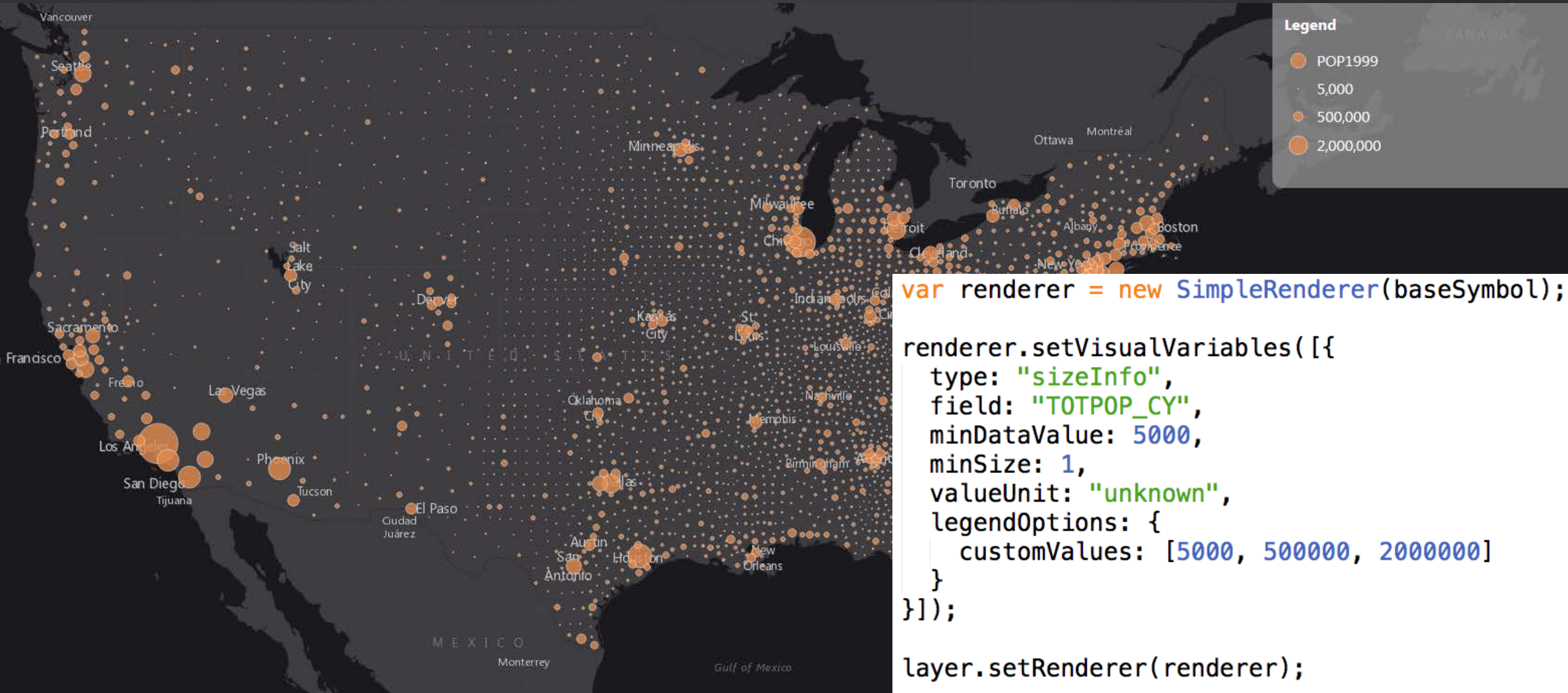
```
renderer.setVisualVariables([  
  {  
    type: "sizeInfo",  
    field: "GroundArea",  
    valueUnit: "feet",  
    valueRepresentation: "area"  
  }  
]);
```

```
layer.setRenderer(renderer);
```



Varying Size – By Data

- SizeInfo visual variable



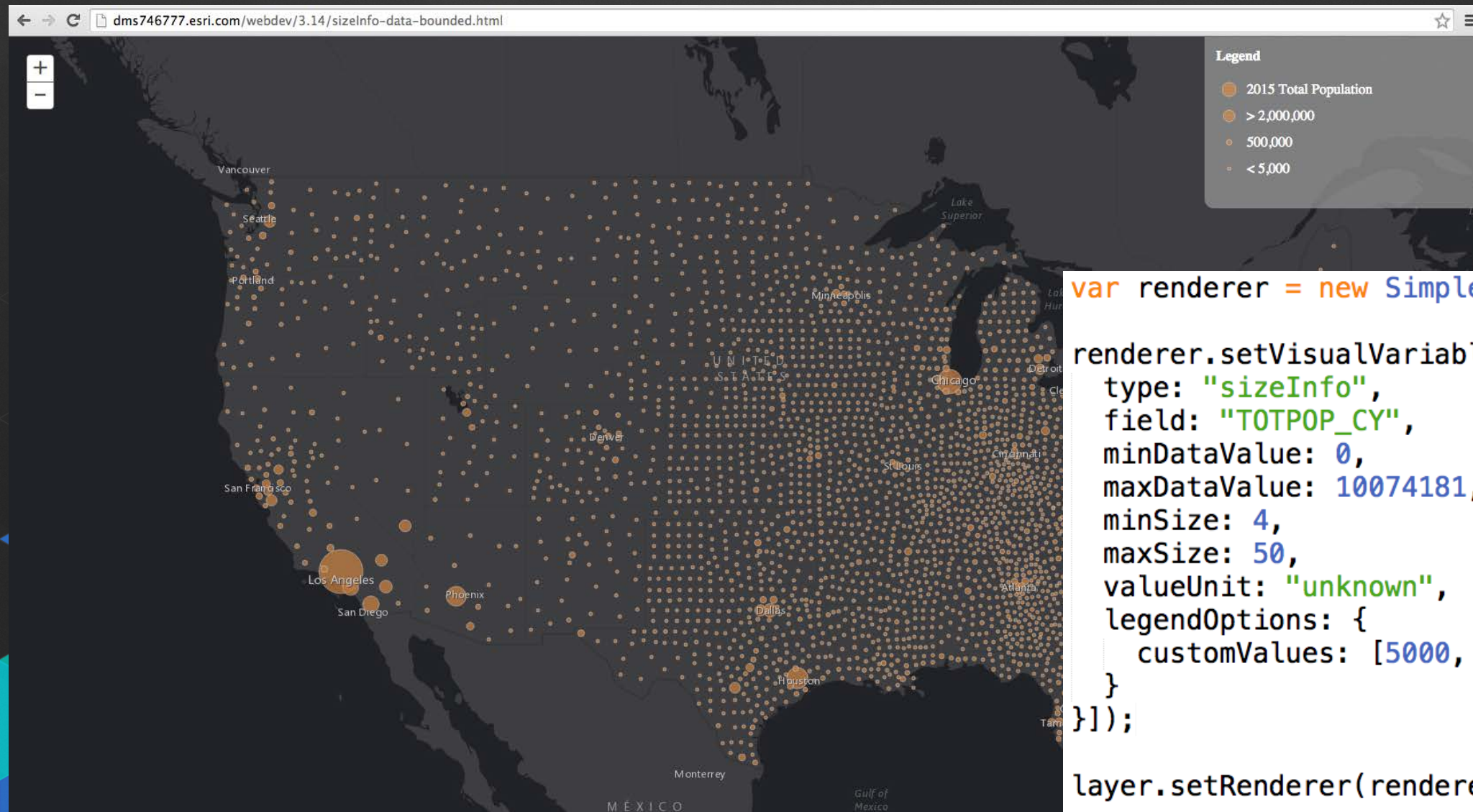
```
var renderer = new SimpleRenderer(baseSymbol);
```

```
renderer.setVisualVariables([{\n  type: "sizeInfo",\n  field: "TOTPOP_CY",\n  minDataValue: 5000,\n  minSize: 1,\n  valueUnit: "unknown",\n  legendOptions: {\n    customValues: [5000, 500000, 2000000]\n  }\n}]);
```

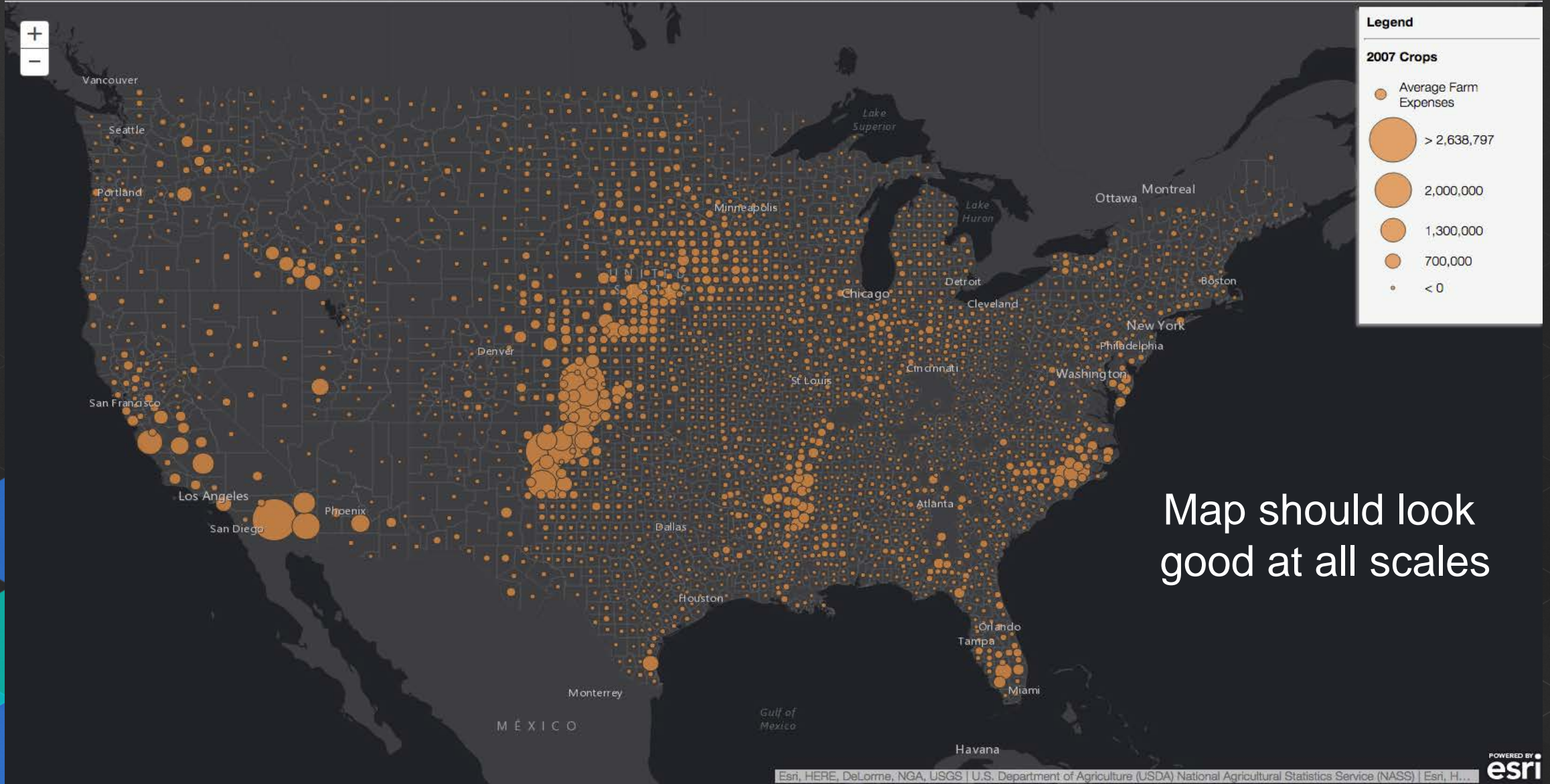
```
layer.setRenderer(renderer);
```


Varying Size – By Data – bounded

- SizeInfo visual variable

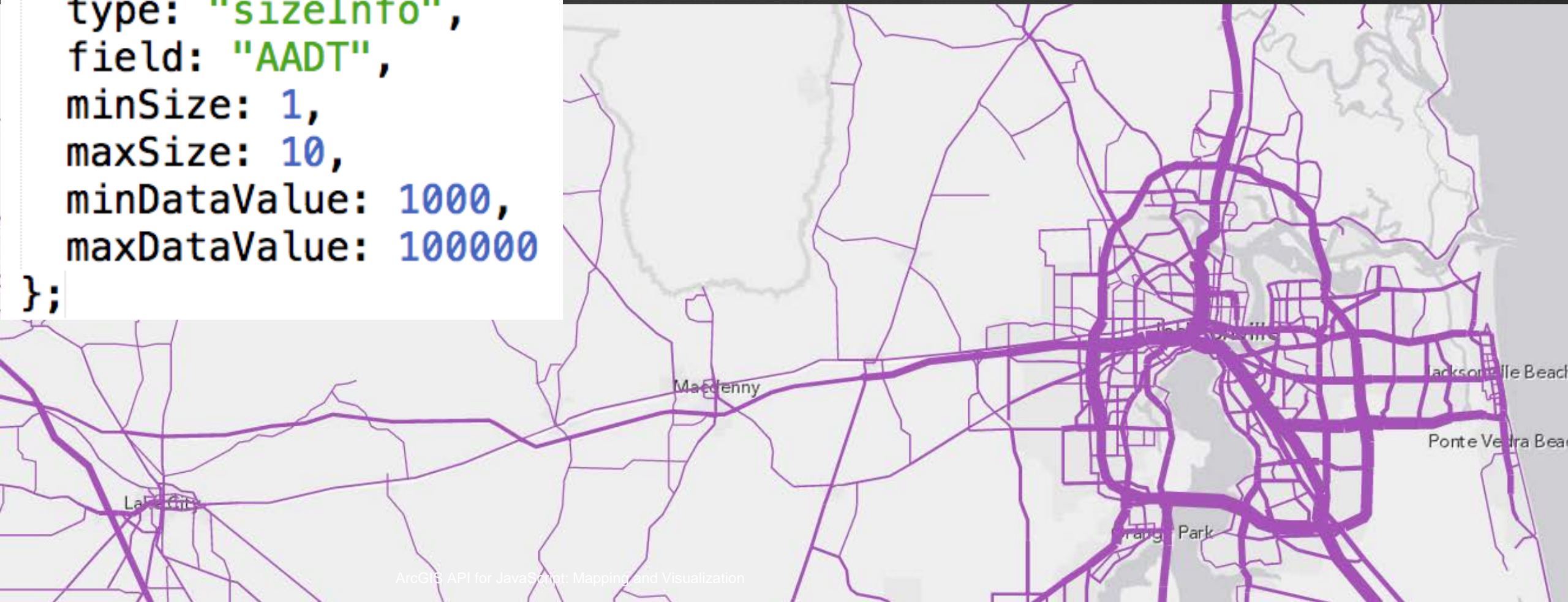


Varying Size – By Data – By Scale



Varying Size – By Value

```
var sizeInfo = {  
  type: "sizeInfo",  
  field: "AADT",  
  minSize: 1,  
  maxSize: 10,  
  minDataValue: 1000,  
  maxDataValue: 100000  
};
```



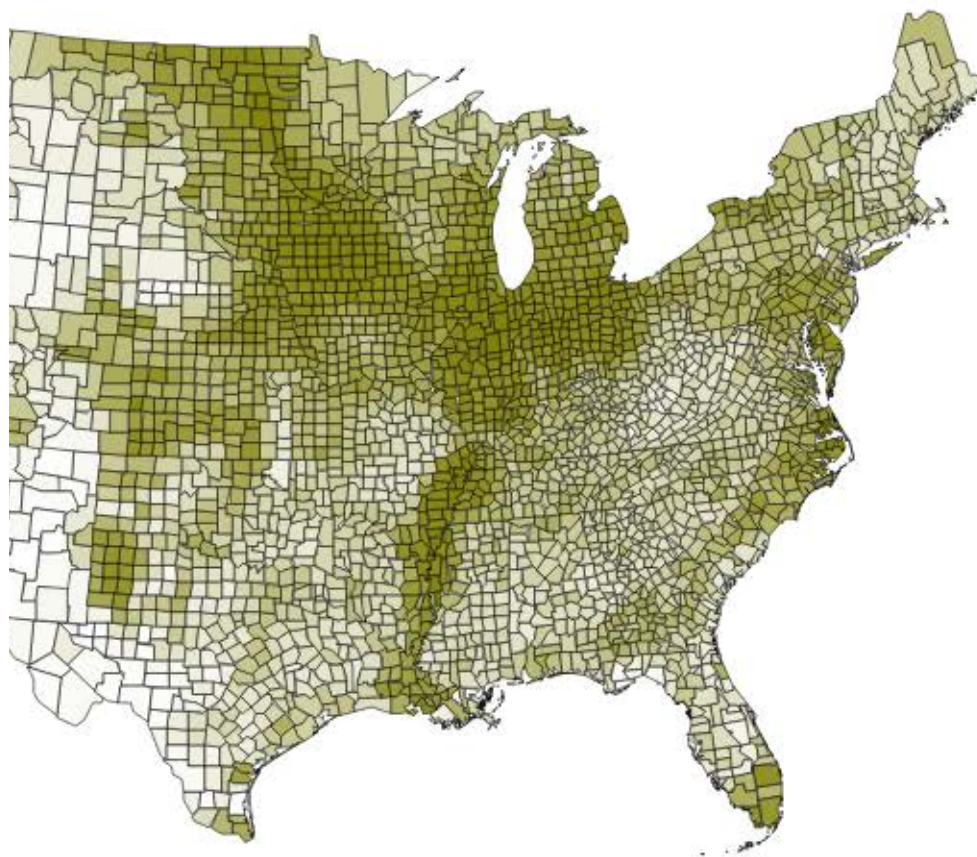
Varying Color by data

- ColorInfo Visual Variable

```
var renderer = new SimpleRenderer();
```

```
renderer.setVisualVariables({  
  type: "colorInfo",  
  field: "M086_07",  
  minDataValue: 0,  
  maxDataValue: 100,  
  colors: [  
    new Color([255, 255, 255]),  
    new Color([127, 127, 0])  
  ]  
});
```

```
layer.setRenderer(renderer);
```

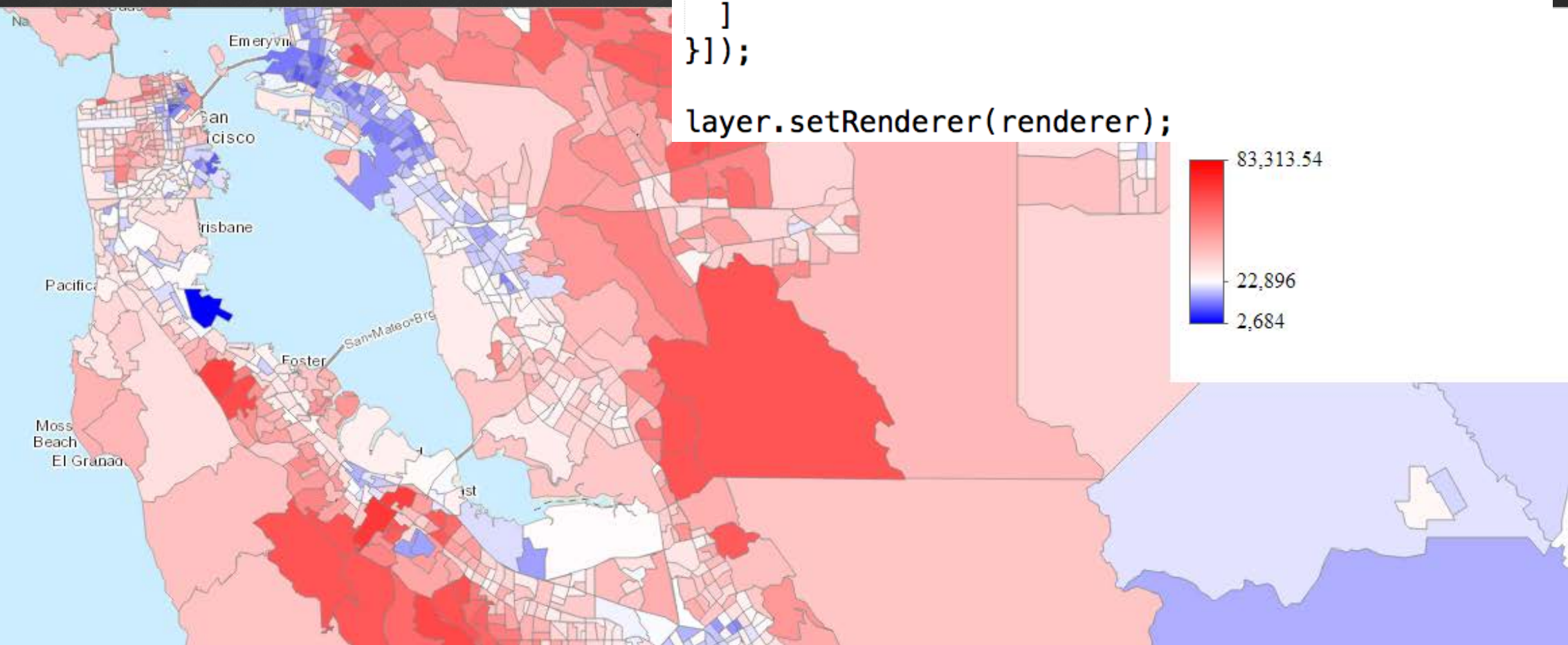


Varying Color with Stops

```
var renderer = new SimpleRenderer();

renderer.setVisualVariables([
  type: "colorInfo",
  field: "X15001_A",
  stops: [
    { value: 2684, color: new Color("blue") },
    { value: 22896, color: new Color("white") },
    { value: 83313, color: new Color("red") }
  ]
});

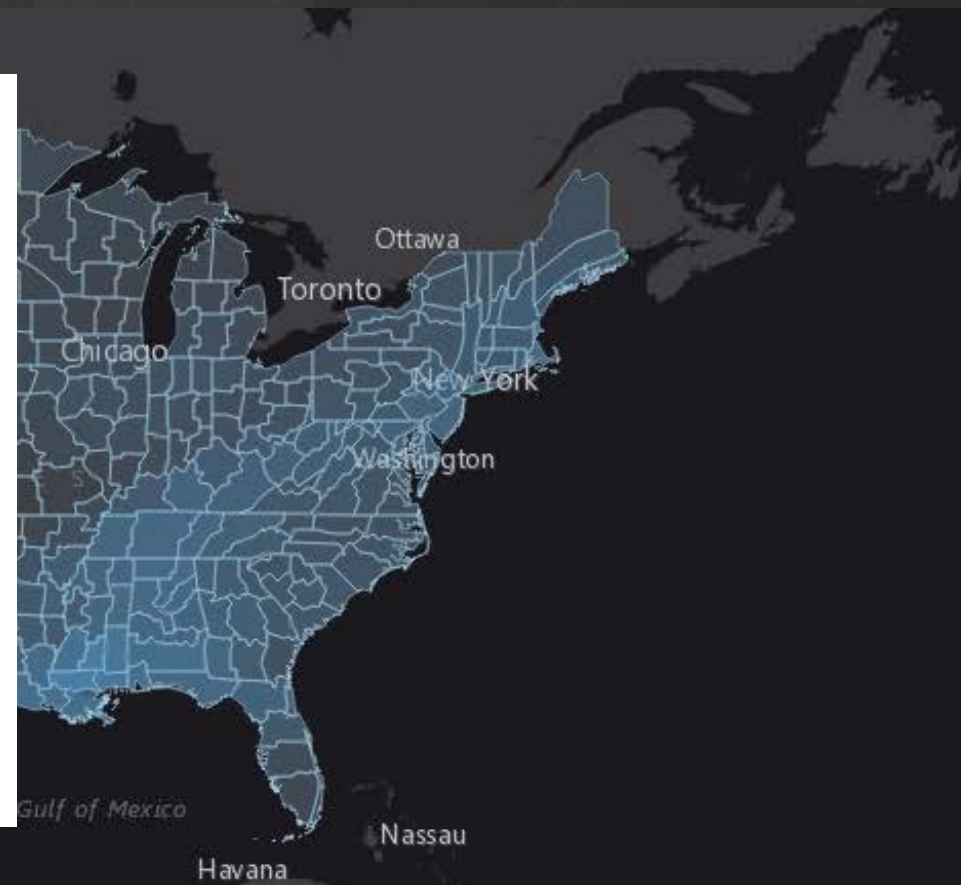
layer.setRenderer(renderer);
```



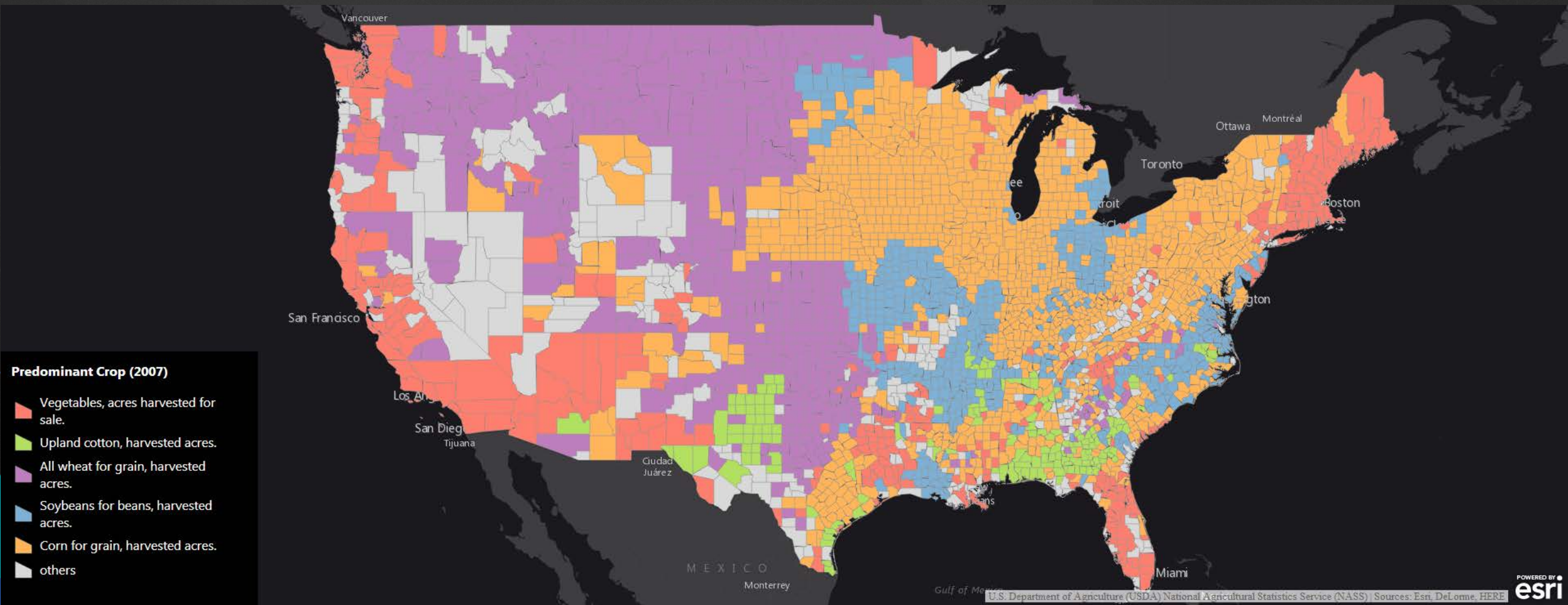
Varying Opacity by data

- OpacityInfo Visual Variable

```
var renderer = new SimpleRenderer();  
  
renderer.symbol.setColor(new Color([76, 141, 188]));  
  
renderer.setVisualVariables([  
  {  
    type: "opacityInfo",  
    field: "PCP",  
    minDataValue: 0,  
    maxDataValue: 10,  
    opacityValues: [0,1]  
  }  
]);  
  
layer.setRenderer(renderer);
```



Predominance Mapping with Transparency

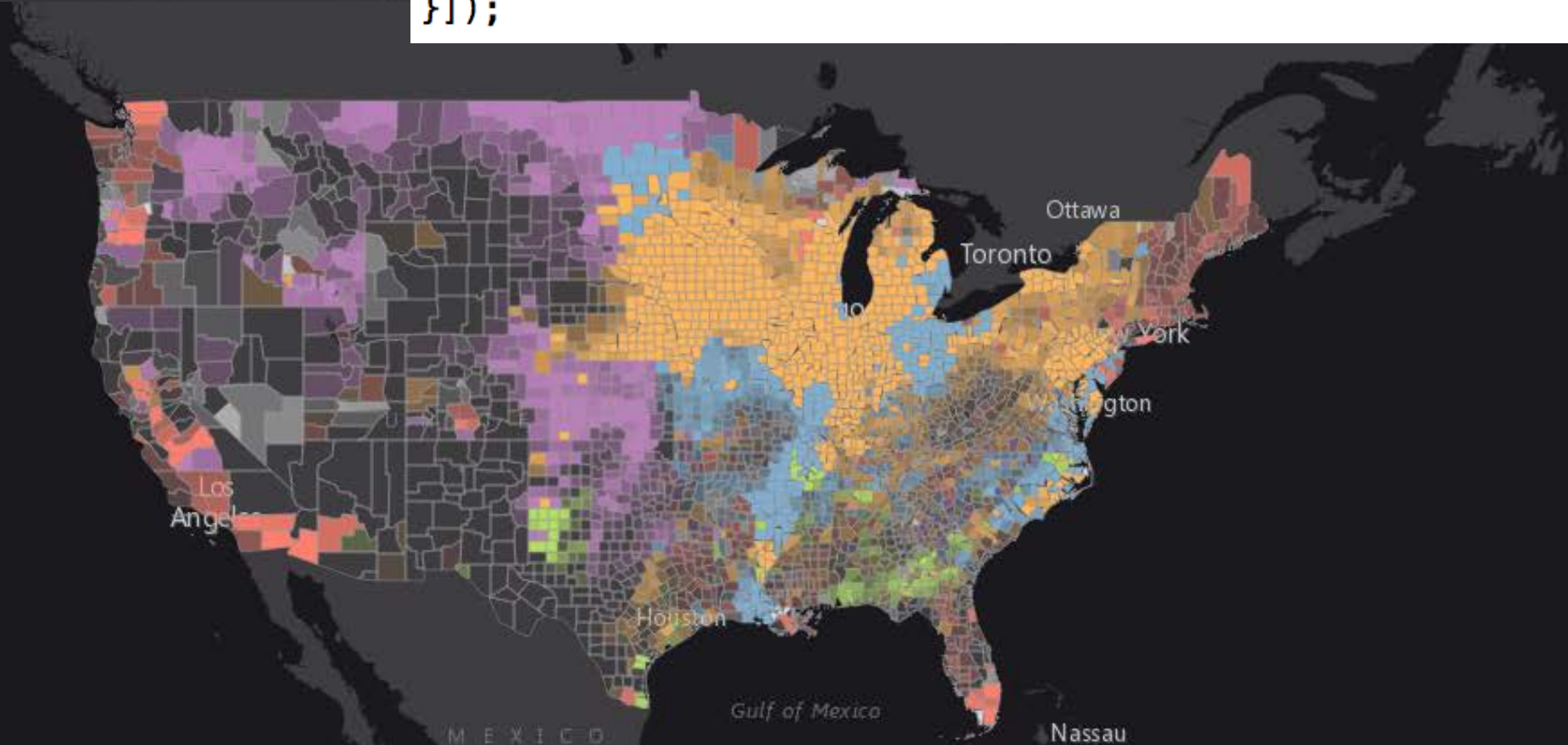


Predominance Mapping with data driven opacity

```
uniqueValueRenderer.setVisualVariables([  
  type: "opacityInfo"  
  field: "M086_07", //percent of farmed land  
  stops: [  
    { value: 10, opacity: 0 }, // -1 stddev,  
    transparent  
    { value: 39, opacity: 0.5 }, // average value, 50%  
    opaque  
    { value: 68, opacity: 1 } // +1 stddev,  
    completely opaque  
  ]  
});
```

Predominant Crop (2007)

- Vegetables, acres harvested for sale.
- Upland cotton, harvested acres.
- All wheat for grain, harvested acres.
- Soybeans for beans, harvested acres.
- Corn for grain, harvested acres.
- others



Renderers to love

DotDensityRenderer

BlendRenderer

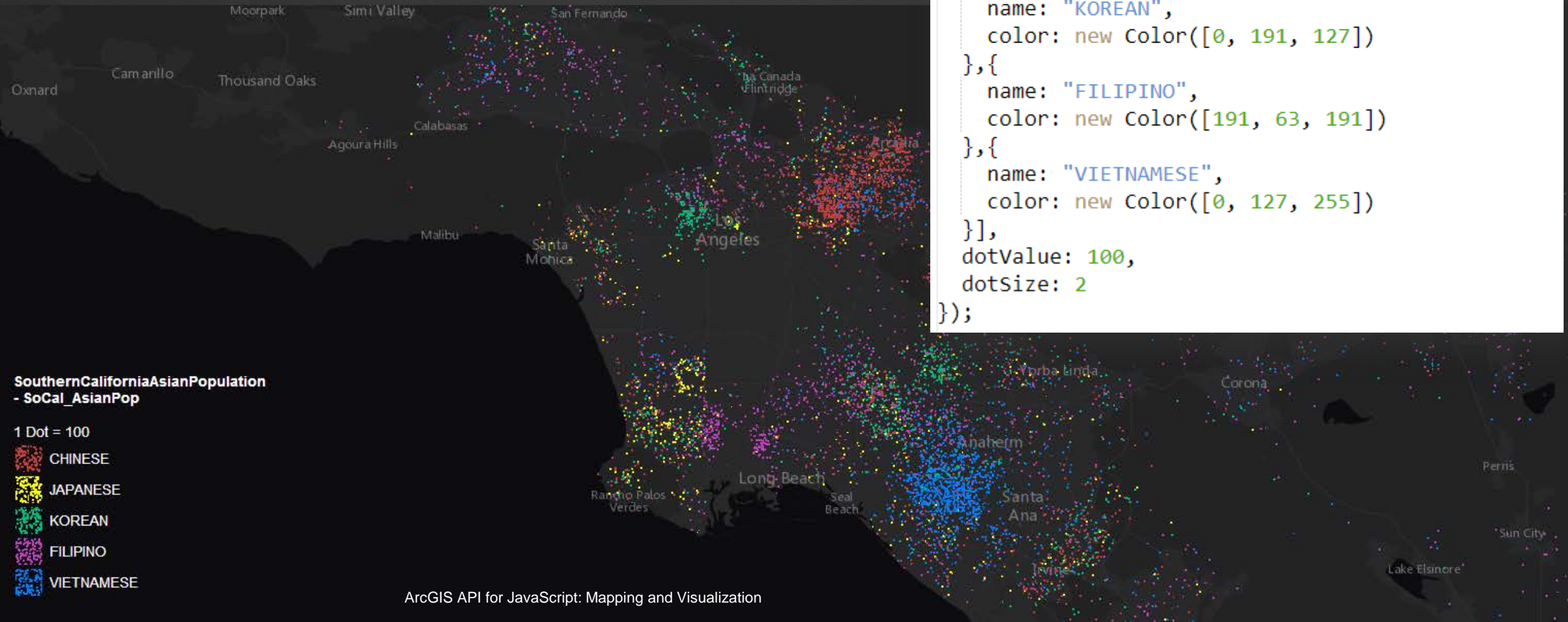
ScaleDependentRenderer

HeatmapRenderer

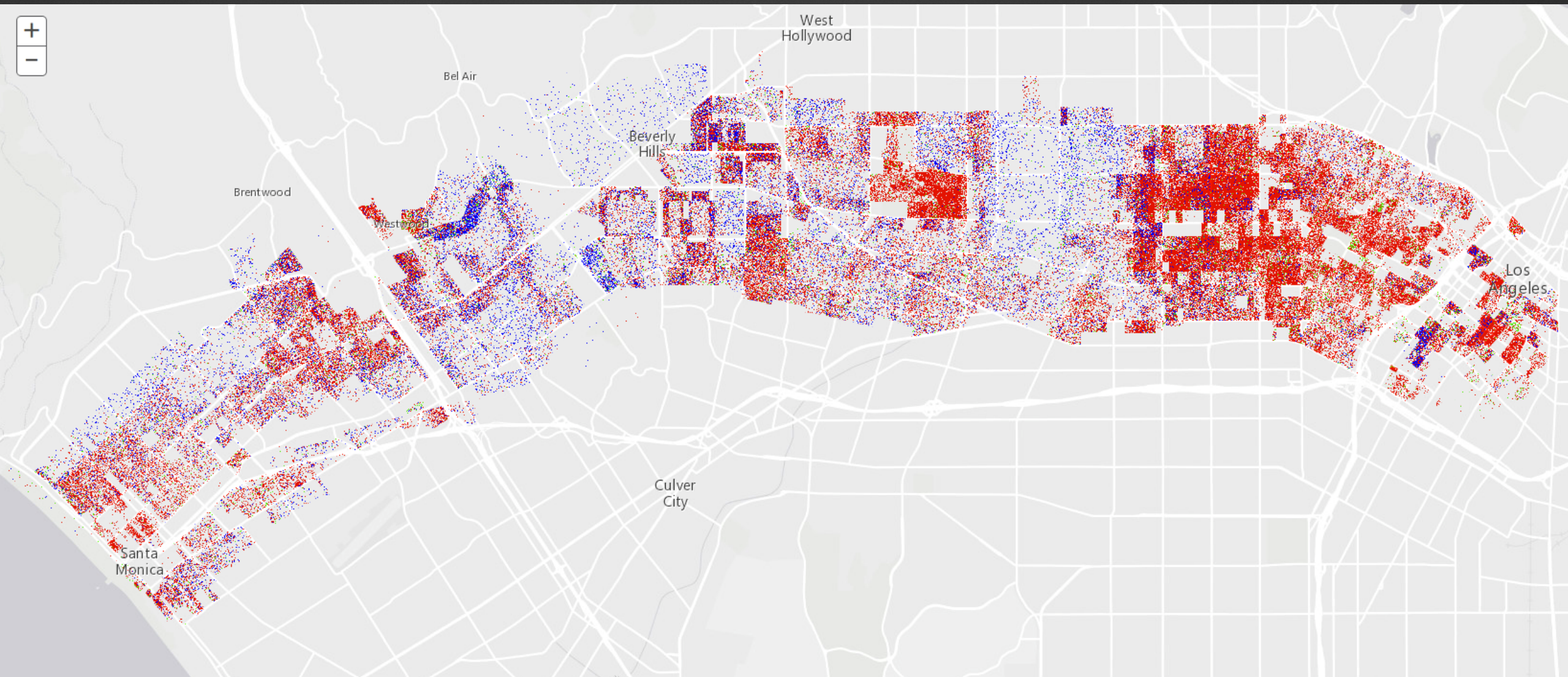
Labeling

Dot Density

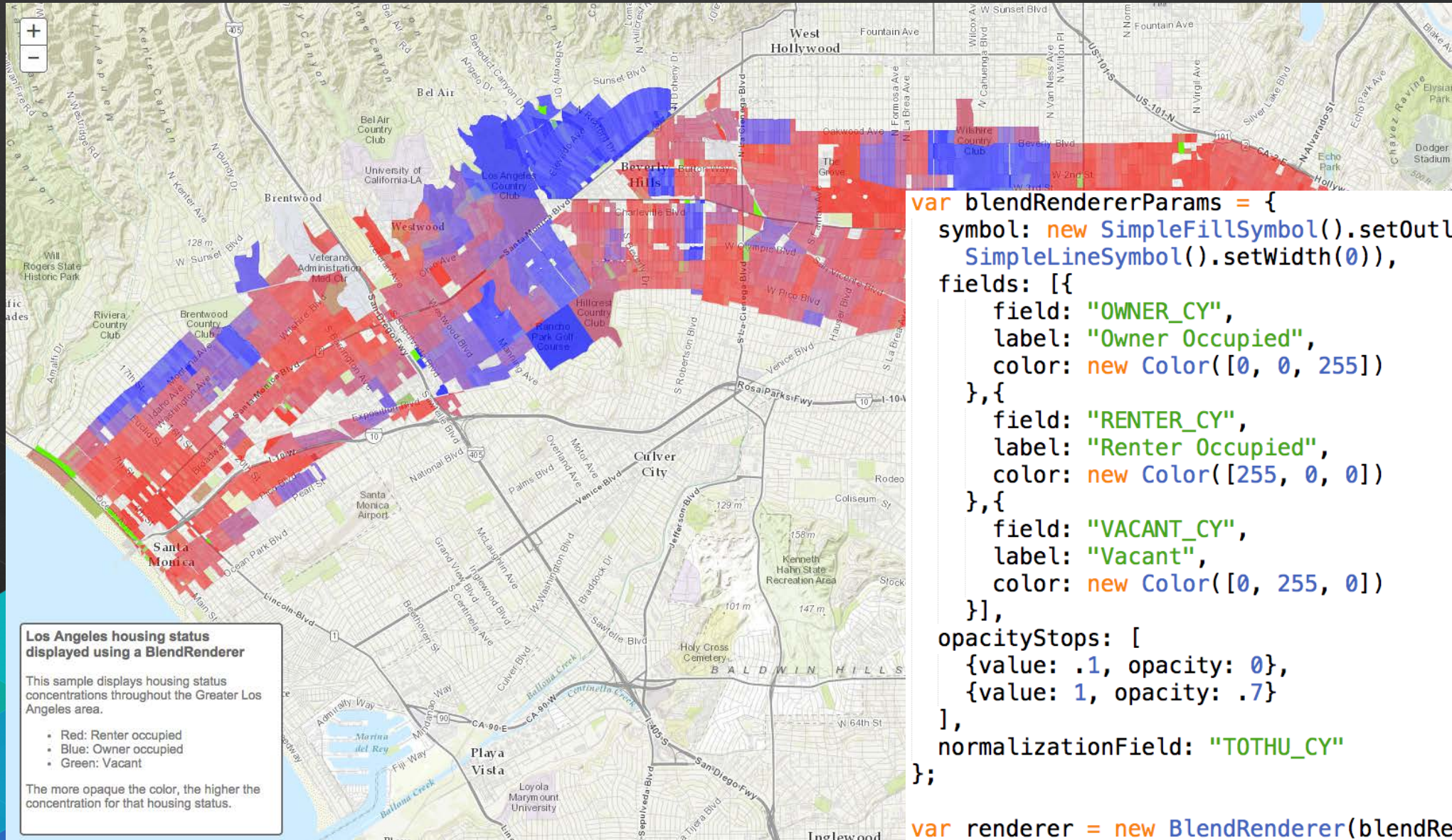
- DotDensityRenderer

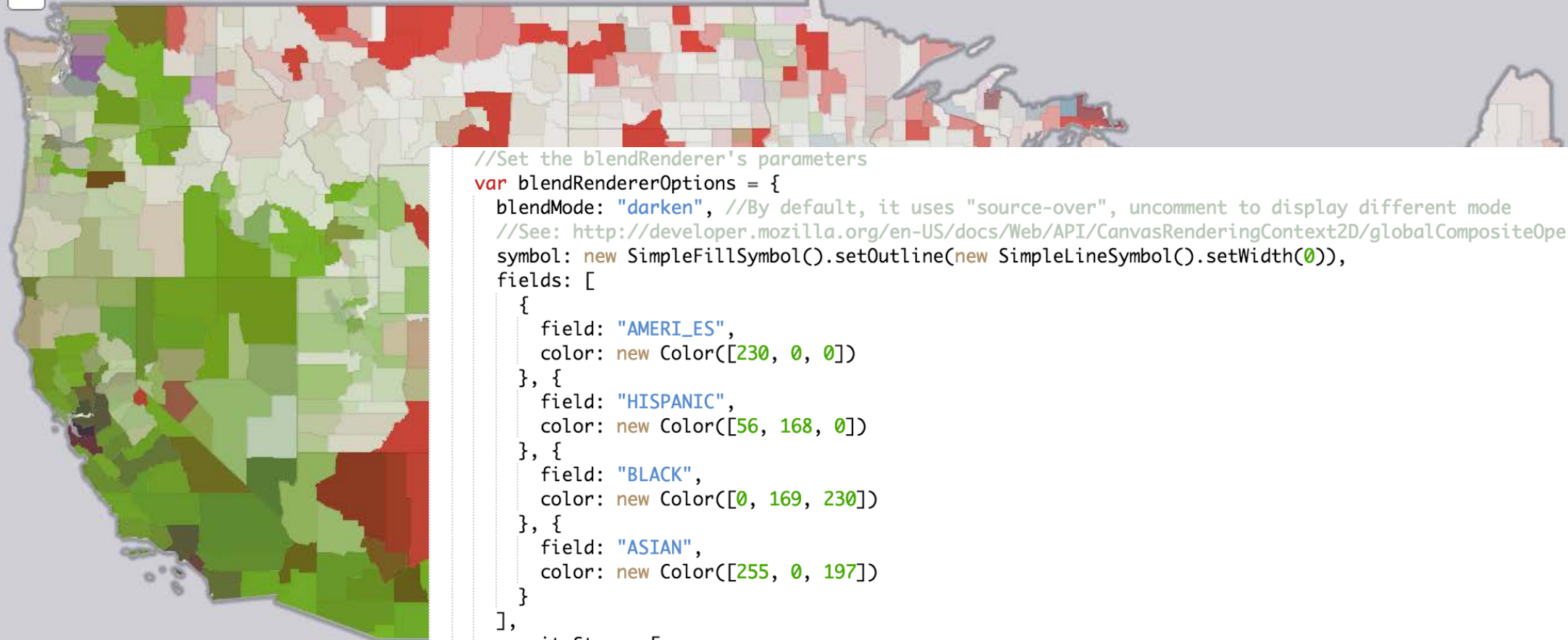


DotDensity can represent 1 dot per value



BlenderRenderer -- Beta





Minority populations displayed using a BlendRenderer

This sample demonstrates minority population concentrations throughout the United States.

- Red: Native American/Alaskan
- Pink/Purple: Asian
- Blue: African American
- Green: Hispanic

The more opaque the color, the higher the concentration for that demographic.

```
//Set the blendRenderer's parameters
var blendRendererOptions = {
  blendMode: "darken", //By default, it uses "source-over", uncomment to display different mode
  //See: http://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/globalCompositeOperation
  symbol: new SimpleFillSymbol().setOutline(new SimpleLineSymbol().setWidth(0)),
  fields: [
    {
      field: "AMERI_ES",
      color: new Color([230, 0, 0])
    }, {
      field: "HISPANIC",
      color: new Color([56, 168, 0])
    }, {
      field: "BLACK",
      color: new Color([0, 169, 230])
    }, {
      field: "ASIAN",
      color: new Color([255, 0, 197])
    }
  ],
  opacityStops: [
    {
      value: 0,
      opacity: 0
    },
    {
      value: .2,
      opacity: .7
    }
  ],
  normalizationField: "POP2012"
};
```

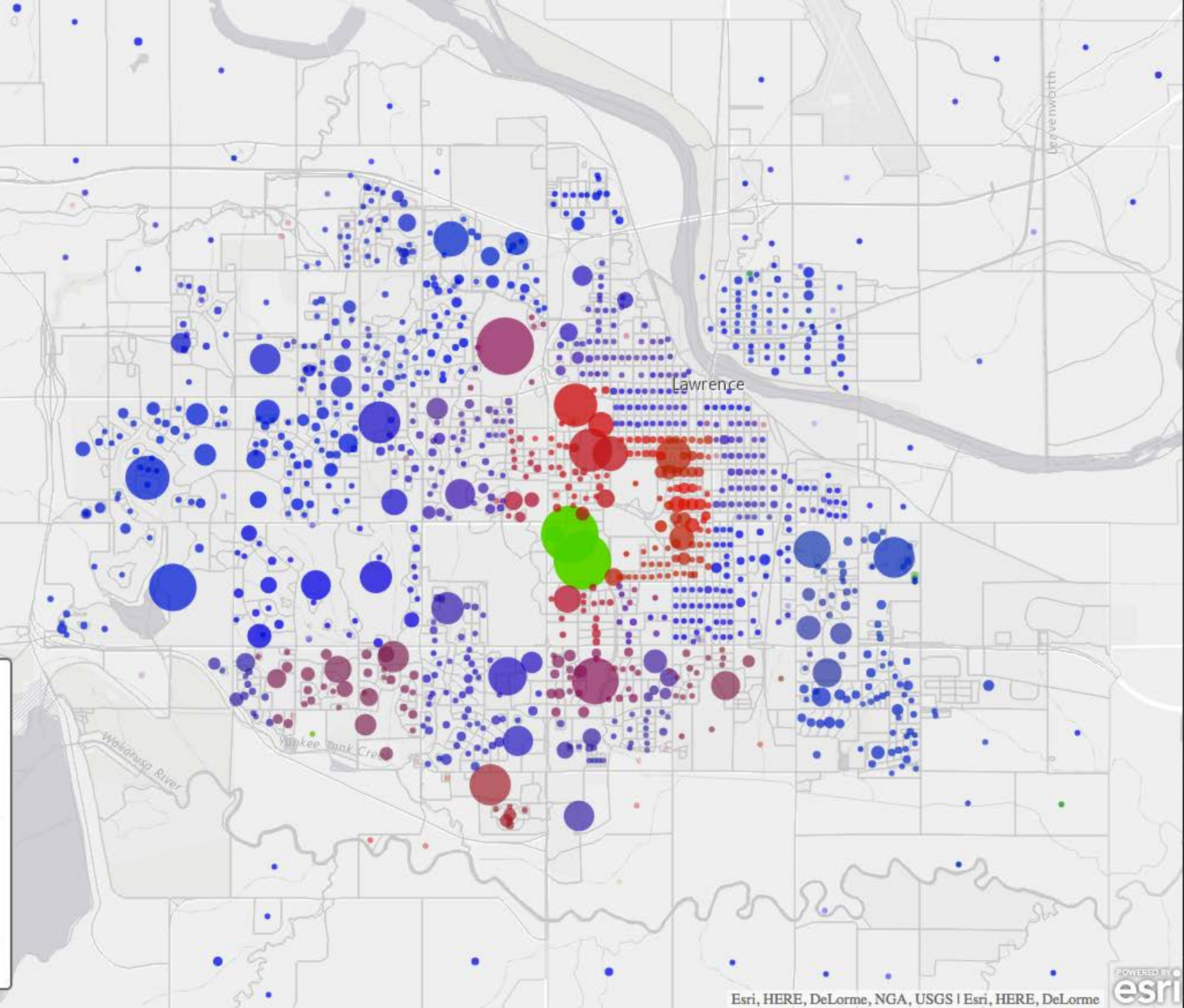


Lawrence, Kansas age groups displayed using a BlendRenderer in addition to total population by size

This sample focuses on Lawrence, KS, where it displays age concentrations with a BlendRenderer. In addition, it also displays the total population in each block using size.

- Red: 20-29
- Blue: 30 and above
- Green: Under 19

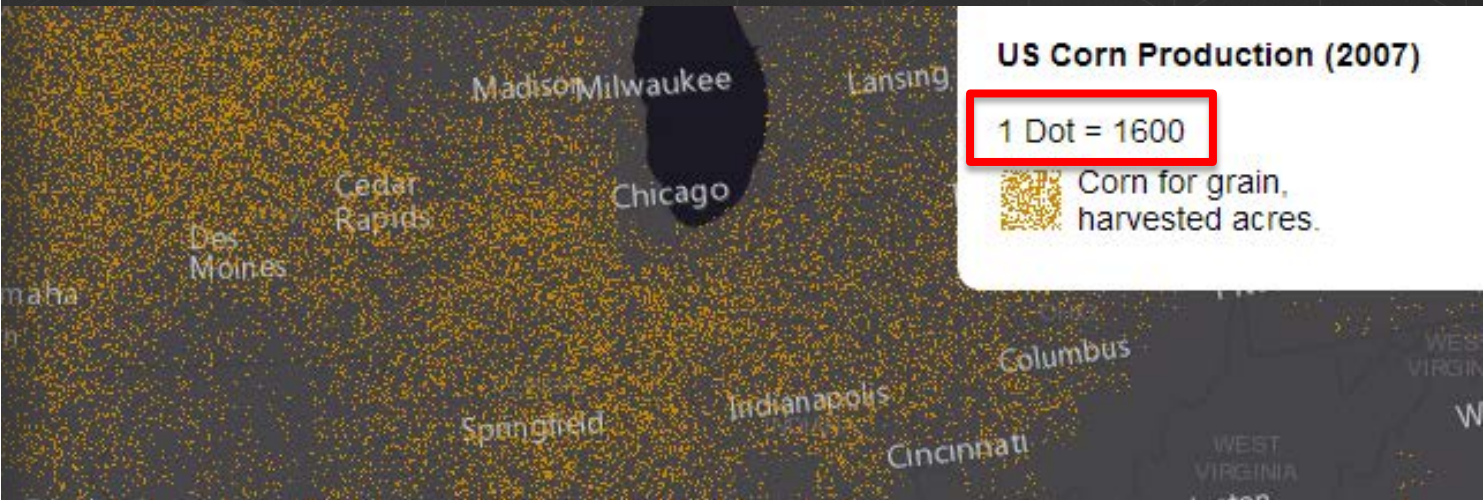
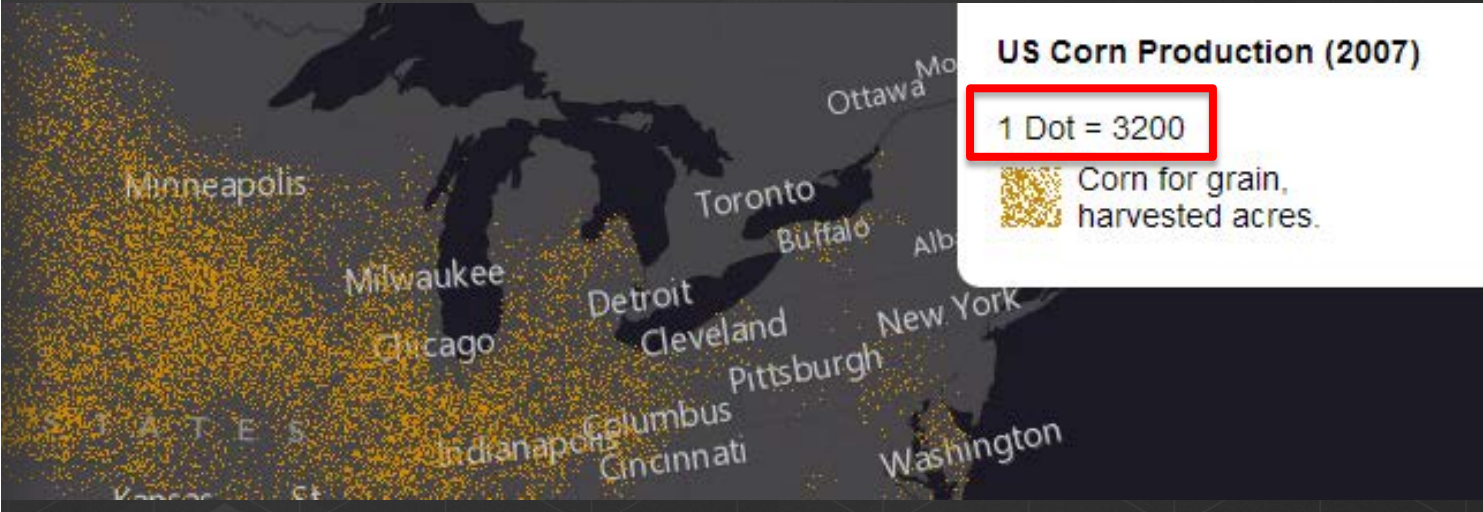
The more opaque the color, the higher the concentration for that age group. In addition, the larger the symbol, the higher the population density for that block.



Scale Dependence

- ScaleDependentRenderer

```
var renderer = new ScaleDependentRenderer({
  rendererInfos: [{
    renderer: new DotDensityRenderer({
      fields: [{
        name: "M163_07",
        color: new Color("#CC8800")
      }],
      dotValue: 3200,
      dotSize: 1
    }),
    maxScale: 17000000,
    minScale: 20000001
  }, {
    renderer: new DotDensityRenderer({
      fields: [{
        name: "M163_07",
        color: new Color("#CC8800")
      }],
      dotValue: 1600,
      dotSize: 1
    }),
    maxScale: 8500000,
    minScale: 17000000
  }
  ]
});
```

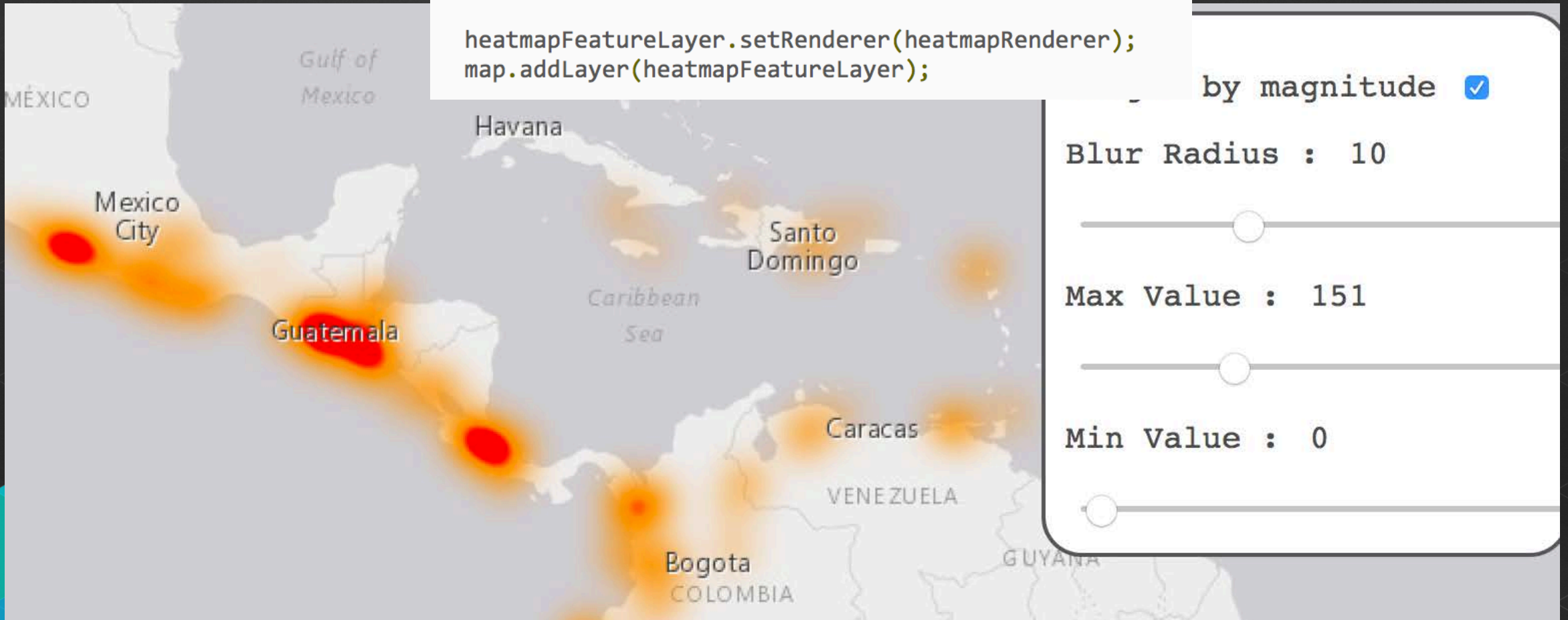


Heatmap

HeatmapRenderer

```
var heatmapRenderer = new HeatmapRenderer({  
  field: "Magnitude",  
  blurRadius: blurCtrl.value,  
  maxPixelIntensity: maxCtrl.value,  
  minPixelIntensity: minCtrl.value  
});
```

```
heatmapFeatureLayer.setRenderer(heatmapRenderer);  
map.addLayer(heatmapFeatureLayer);
```



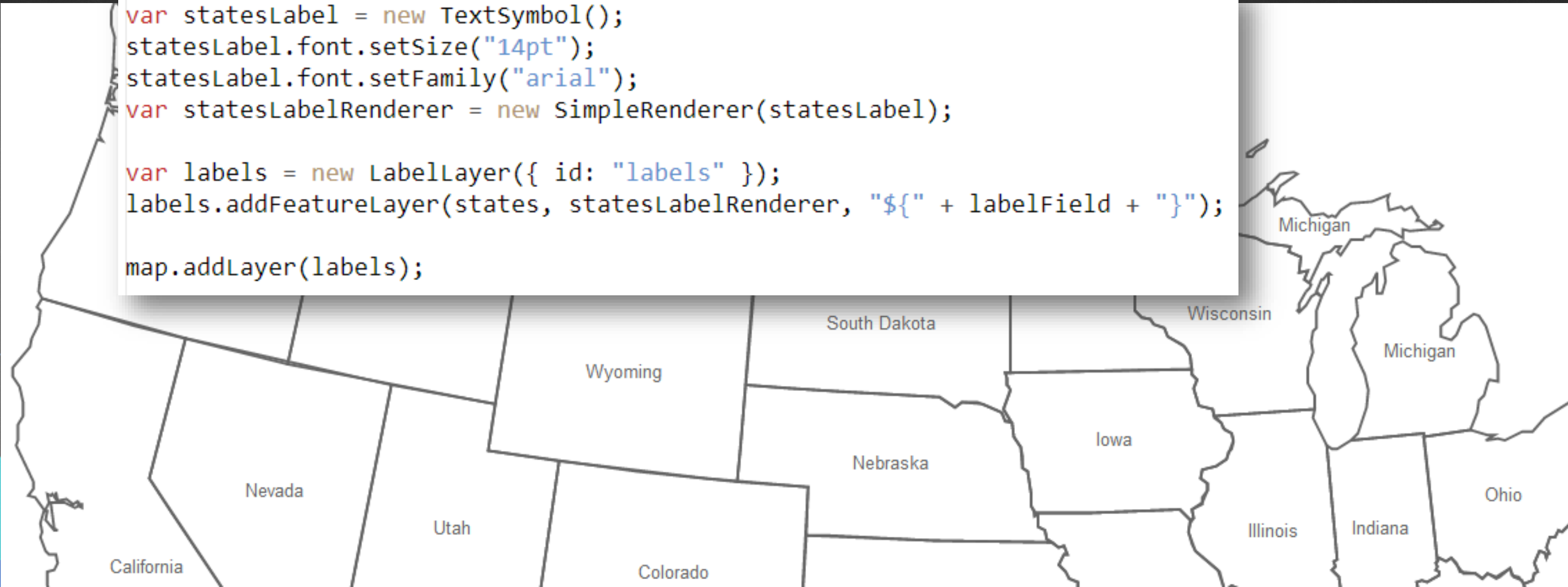
Labeling

- LabelLayer

```
var statesLabel = new TextSymbol();
statesLabel.font.setSize("14pt");
statesLabel.font.setFamily("arial");
var statesLabelRenderer = new SimpleRenderer(statesLabel);

var labels = new LabelLayer({ id: "labels" });
labels.addFeatureLayer(states, statesLabelRenderer, "${" + labelField + "}");

map.addLayer(labels);
```



Let your data drive your visualization

SmartMapping

`createColorRenderer`

`createSizeRenderer`

`createTypeRenderer`

`createPredominanceRenderer`

Smart Mapping

- Data driven defaults
 - Introduced in 3.13
 - Takes the guesswork out of:
 - What colors to use...for fill, outline, background fill outline, etc
 - What opacity to use on each color
 - What size to use for lines, markers, outlines on markers
 - What size is appropriate for what scale
 - Does this by looking at the data, the basemap, and the story you want to tell
- Used by authoring and data exploration apps

Smart Mapping API components -- Styles

Object: `esri/styles/choropleth`

[[AMD Module Require](#) | [Legacy Module Require](#)]

```
require(["esri/styles/choropleth"], function(esriStylesChoropleth) { /* code goes here */ });
```

Description

(Added at v3.13)

This module contains a collection of themes suitable for unclassed and classed choropleth mapping.

```
require([
  "esri/map",
  "esri/styles/choropleth", "dojo/domReady!"
], function (Map, esriStylesChoropleth){
  map = new Map("map", {
    basemap: "topo",
    center: [-122.45, 37.75], // longitude, latitude
    zoom: 13
  });

  map.on("load", function (){
    schemes = esriStylesChoropleth.getSchemes({theme: "high-to-low", basemap: map.getBasemap(), geometryType: "point"});
    console.log(JSON.stringify(schemes));
  });
});
```


Smart Mapping API components – FL Stats Plugin

Class: FeatureLayerStatistics

[[AMD Module Require](#) | [Legacy Module Require](#)]

```
require(["esri/plugins/FeatureLayerStatistics"], function(FeatureLayerStatistics) { /* code goes here */ });
```

Description

(Added at v3.13)

This module defines a class and a feature layer plugin that is used to calculate feature layer statistics. The return value of this module is a class, but functions that allows it to be added as a plugin to a feature layer. When this module is added as a plugin, it adds an instance of the class as a new plugin to the feature layer called stats.

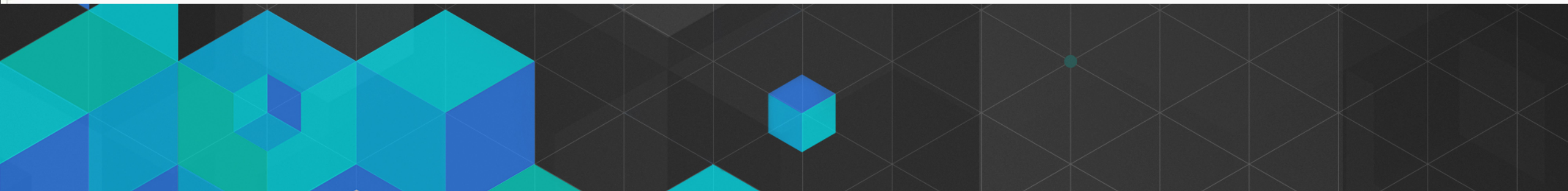
```
require([ "esri/layers/FeatureLayer" ], function(FeatureLayer) {  
  
  var featureLayer = new FeatureLayer("//services.arcgis.com/V6ZHFr6zdgNZuVG0/ArcGIS/rest/services/USA_Counties/FeatureServer/0");  
  
  featureLayer  
    .addPlugin("esri/plugins/FeatureLayerStatistics")  
    .then(function() {  
      featureLayer  
        .statisticsPlugin  
        .getUniqueValues({ field: "STATE_NAME" })  
        .then(function(result) {  
          console.log("Unique values: ", result.uniqueValueInfos);  
        });  
    });  
});
```

Smart Mapping API components – FL Stats Plugin

```
require(["esri/plugins/FeatureLayerStatistics"], function(FeatureLayerStatistics) { /* code goes here */ });
```

Methods

Name	Return type	Summary
<code>add(layer, options?)</code>	None	This function is called internally when the plugin is added to a feature layer.
<code>getClassBreaks(params)</code>	Promise	Calculate class breaks for data stored in the given field.
<code>getFieldStatistics(params)</code>	Promise	Calculate basic statistics for data stored in the given field.
<code>getHeatmapStatistics(options?)</code>	Promise	Calculate heatmap statistics.
<code>getHistogram(params)</code>	Promise	Calculate histogram for data stored in the given field.
<code>getSampleFeatures(options?)</code>	Promise	Get a random sampling of features in this layer.
<code>getSuggestedScaleRange(options?)</code>	Promise	Find optimal scale range for viewing this layer.
<code>getUniqueValues(params)</code>	Promise	Find unique values available for the given field.



Smart Mapping API components – smartMapping

Object: [esri/renderers/smartMapping](#)

[Print this page](#)

[[AMD Module Require](#) | [Legacy Module Require](#)]

[Methods](#)

```
require(["esri/renderers/smartMapping"], function(smartMapping) { /* code goes here */ });
```

Description

(Added at v3.13)

This module contains a collection of helper functions used to create pre-configured renderers for smart feature styling.

Samples

Search for [samples](#) that use this class.

Methods

Name	Return type	Summary
createClassColorRenderer(params)	Promise	Creates a renderer for visualizing features using colors.
createClassSizeRenderer(params)	Promise	Creates a renderer for visualizing features by varying their size.
createClassColorRenderer(params)	Promise	Creates a renderer for visualizing features using colors.
createHeatmapRenderer(params)	Promise	Creates a renderer for visualizing features using heatmap.
createOpacityInfo(params)	Promise	Creates an object that describes how opacity of features is calculated.
createClassSizeRenderer(params)	Promise	Creates a renderer for visualizing features by varying their size based on data.
createClassTypeRenderer(params)	Promise	Creates a renderer for visualizing features by their type.

Smart Mapping API components – smartMapping

createColorRenderer(params)

Creates a renderer for visualizing features using colors. Colors are picked from a color gradient by mapping a feature's data attribute (Example: population) to a specific color in the gradient. See the **Object Specifications** table below for the structure of the params object and [Promise](#).

Return type: [Promise](#)

Parameters:

<Object> params	Required	See the object specifications table below for the structure of the params object.
---------------------------------------	----------	---

Object Specifications:

[<Promise>](#)

<Renderer> renderer	A pre-configured ClassBreaksRenderer with a visual variable for colorInfo. Use FeatureLayer.setRenderer to apply this renderer to a layer.
<Object> statistics	Field statistics (min, max, avg, stddev) used to create the renderer.

[<params>](#)

<String> basemap	Basemap used for your map. The following basemaps are supported: streets, gray, topo, terrain, national-geographic, oceans, osm, satellite, hybrid, dark-gray. If you're not using any of these basemaps, then pick one that closely resembles your basemap.
<String> field	Name of the attribute field that contains data values.
<FeatureLayer> layer	Feature layer for which this renderer is created.
<Object> scheme	Optional: An object describing the style scheme used for the renderer. A style scheme can be obtained by calling esri/styles/choropleth.getSchemes method – either the primary scheme or a secondary scheme can be used. In the absence of scheme, both theme and basemap should be specified.
<Boolean> showOthers	Optional: Indicates whether the renderer should display features that don't have any value for the specified field. Default is true.
<Object> statistics	Optional: Field statistics (min, max, avg, stddev) used to create the renderer. If not provided, FeatureLayerStatistics.getFieldStatistics will be used to calculate the statistics.
<String> theme	Optional: Theme to be used. The following themes are supported: high-to-low, above-and-below, centered-on, extremes and group-similar. Default is high-to-low.

SmartMapping in the ArcGIS API for JavaScript

It's all there

Object: `esri/renderers/smartMapping`

[[AMD Module Require](#) | [Legacy Module Require](#)]

[Methods](#)

```
require(["esri/renderers/smartMapping"], function(smartMapping) { /* code goes here */ });
```

Description

(Added at v3.13)

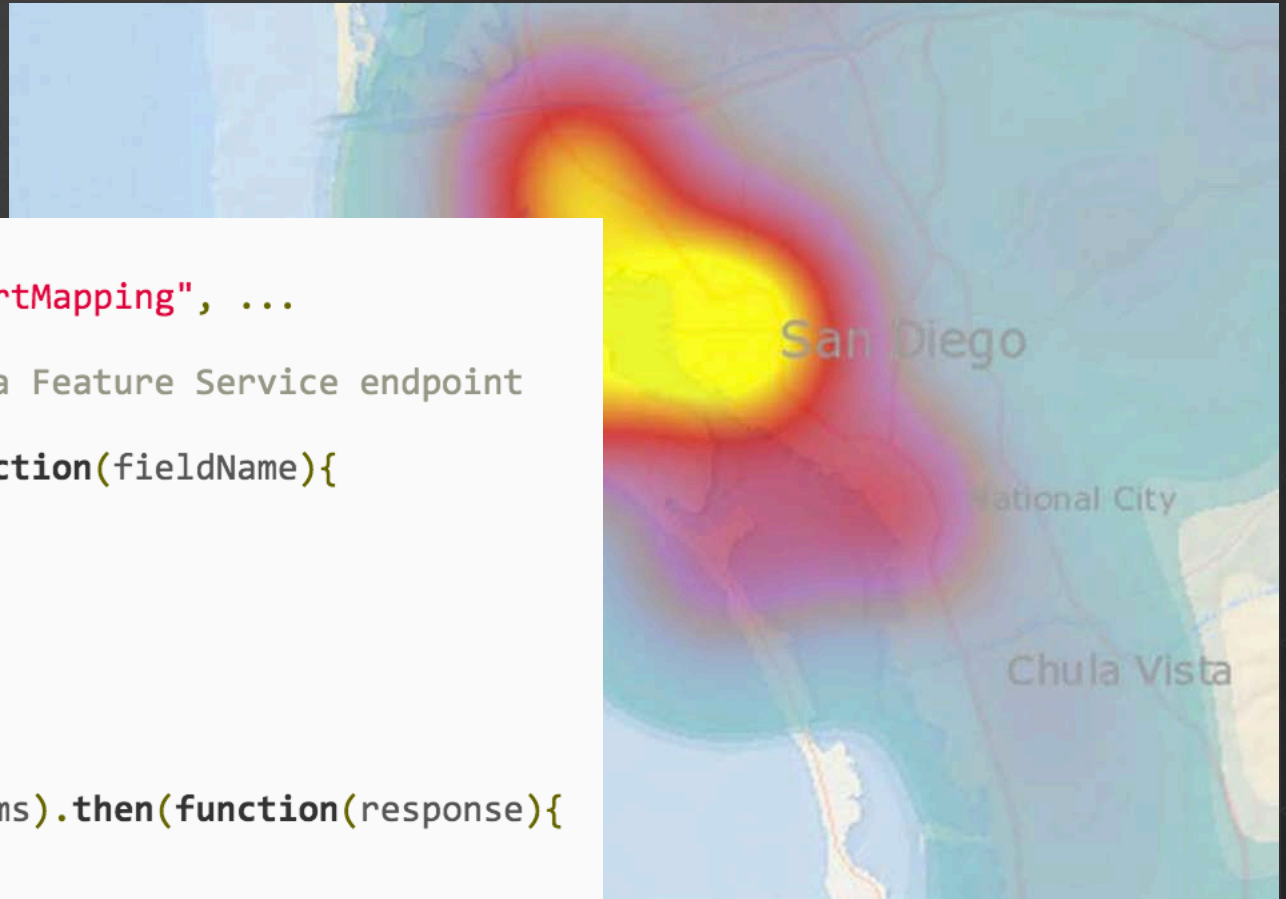
This module contains a collection of helper functions used to create pre-configured renderers for smart feature styling.

```
smartMapping.createColorRenderer({  
  layer: featureLayer,  
  field: "AvgIncome",  
  basemap: "topo",  
  theme: "high-to-low"  
}).then(function (colorRenderer){  
  featureLayer.setRenderer(colorRenderer.renderer);  
});
```

smartMapping Heatmap

createHeatmapRenderer

```
require([
  "esri/layers/FeatureLayer", "esri/renderers/smartMapping", ...
], function(FeatureLayer, smartMapping, ... ) {
  var fl = new FeatureLayer( ... ); //points to a Feature Service endpoint
  var params = { layer: fl };
  smartMapping.getSuggestedField(params).then(function(fieldName){
    var params = {
      basemap: "gray",
      field: fieldName,
      layer: fl
    };
    return params;
  }).then(function(rendererParams){
    smartMapping.createColorRenderer(rendererParams).then(function(response){
      fl.setRenderer(response.renderer);
      fl.redraw();
    });
  });
});
```



smartMapping Color createColorRenderer

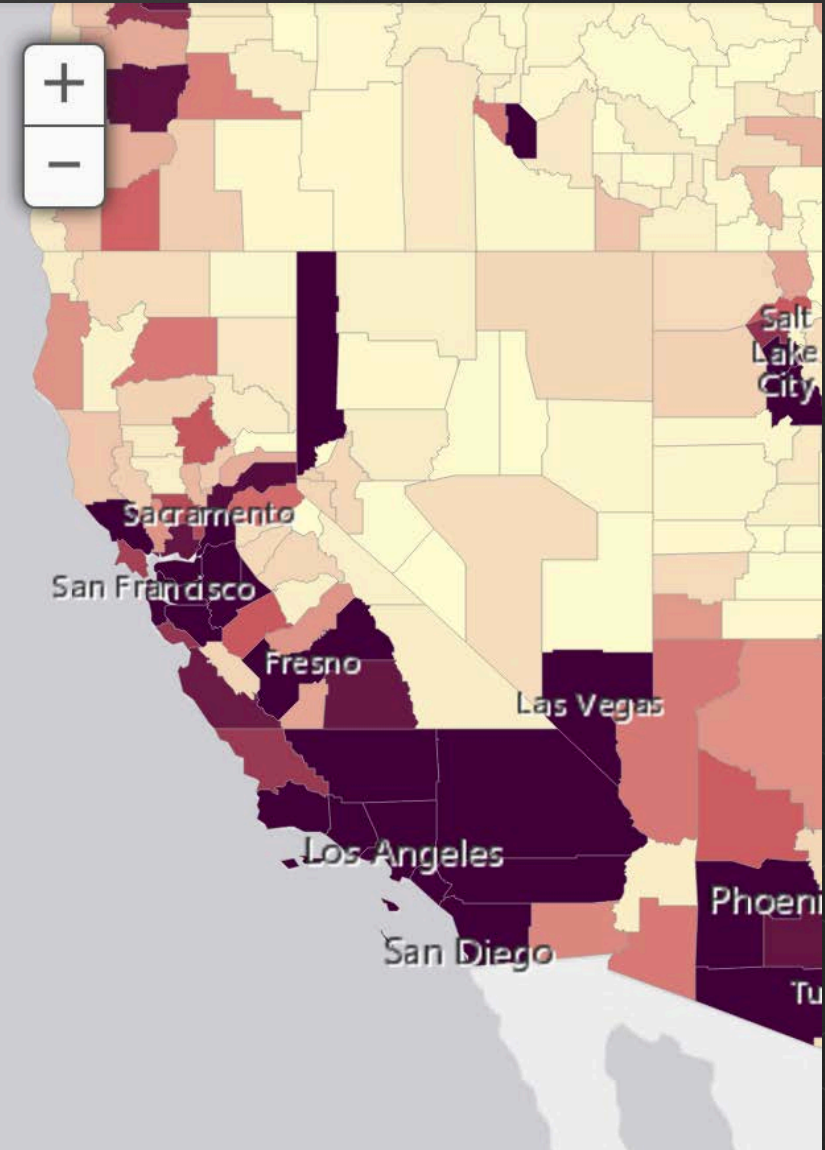
```
// -----  
// Calculate the Histogram  
// -----  
featureLayerStatistics.getHistogram({  
  field: fieldName,  
  numBins: 10  
}).then(function (histogram){  
  //console.log("histogram is created", histogram);  
  // -----  
  // Update the ColorInfoSlider and apply FeatureLayerStatistics histogram  
  // -----  
  var sliderHandleInfo = getSliderHandlePositions(theme);  
  colorInfoSlider.set("colorInfo", colorRenderer.renderer.visualVariables[0]);  
  colorInfoSlider.set("minValue", colorRenderer.statistics.min);  
  colorInfoSlider.set("maxValue", colorRenderer.statistics.max);  
  colorInfoSlider.set("statistics", colorRenderer.statistics);  
  colorInfoSlider.set("histogram", histogram);  
  colorInfoSlider.set("handles", sliderHandleInfo["handles"]);  
  colorInfoSlider.set("primaryHandle", sliderHandleInfo["primaryHandle"]);  
  busyIndicator.hide();  
  
  // -----  
  // process slider handle changes  
  // Object with keys: type, field, normalizationField, stops  
  // -----  
  colorInfoSlider.on("handle-value-change", function (sliderValueChange){  
    //console.log("handle-value-change", sliderValueChange);  
    geoenrichedFeatureLayer.renderer.setVisualVariables([sliderValueChange]);  
    geoenrichedFeatureLayer.redraw();  
  });  
});
```

Movie Genre by County

plugin

renderer);

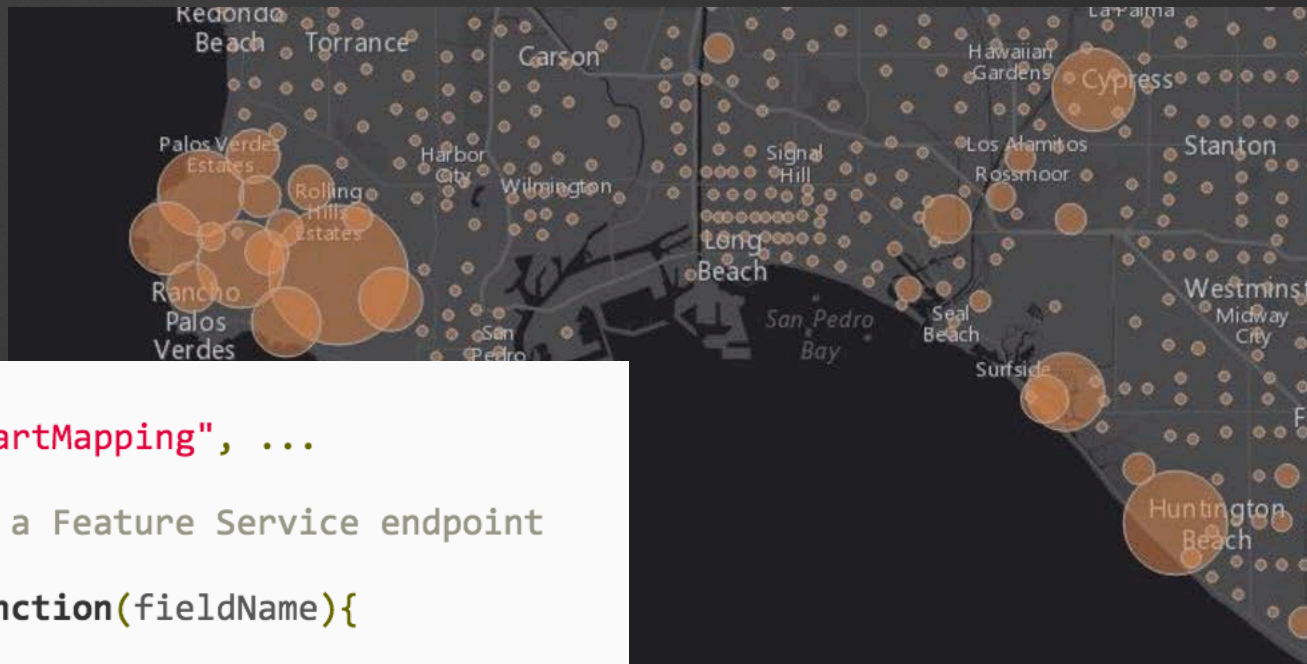
all



smartMapping Size

createSizeRenderer

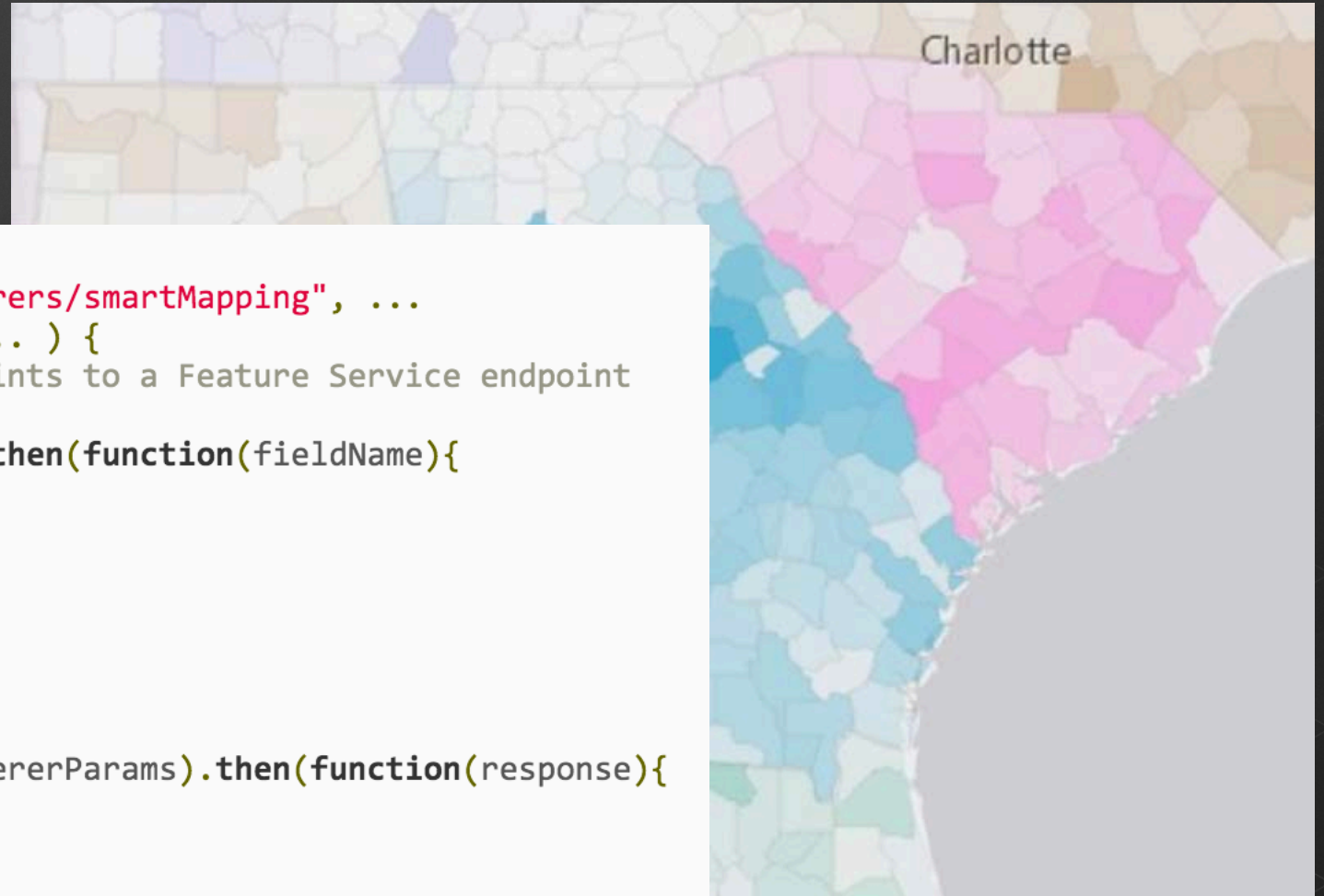
```
require([
  "esri/layers/FeatureLayer", "esri/renderers/smartMapping", ...
], function (FeatureLayer, smartMapping, ... ) {
  var fl = new FeatureLayer( ... ); //points to a Feature Service endpoint
  var params = { layer: fl };
  smartMapping.getSuggestedField(params).then(function(fieldName){
    var params = {
      basemap: "gray",
      field: fieldName,
      layer: fl
    };
    return params;
  }).then(function(rendererParams){
    smartMapping.createColorRenderer(rendererParams).then(function(response){
      fl.setRenderer(response.renderer);
      fl.redraw();
    });
  });
});
```



smartMapping Type

createTypeRenderer

```
require([
  "esri/layers/FeatureLayer", "esri/renderers/smartMapping", ...
], function(FeatureLayer, smartMapping, ... ) {
  var fl = new FeatureLayer( ... ); //points to a Feature Service endpoint
  var params = { layer: fl };
  smartMapping.getSuggestedField(params).then(function(fieldName){
    var params = {
      basemap: "gray",
      field: fieldName,
      layer: fl
    };
    return params;
  }).then(function(rendererParams){
    smartMapping.createColorRenderer(rendererParams).then(function(response){
      fl.setRenderer(response.renderer);
      fl.redraw();
    });
  });
});
```



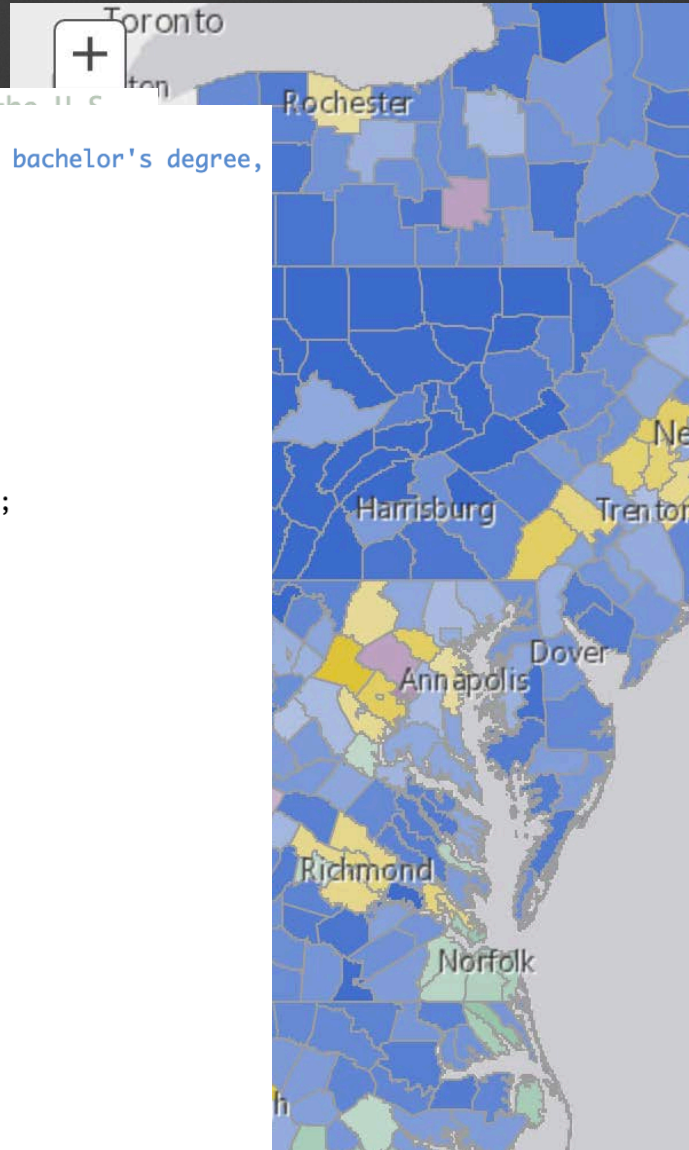
smartMapping Predominance

createPredominanceRenderer

```
//Creates the predominance renderer
vc } else if (dropdown.value === "postgrad"){
    legend.layerInfos[0].title = "Of the adults (25+) who completed a bachelor's degree,
    params = postGradParams;
  } else {
    console.log("Not a valid menu selection");
    return;
  }
  params.layer = edLayer;
  params.basemap = map.getBasemap();
  params.includeOpacityInfo = true;

  //Creates the predominance renderer
  smartMapping.createPredominanceRenderer(params).then(applyRenderer);
}

//Create a legend
var legend = new Legend({
  map: map,
  layerInfos: [{
    layer: edLayer,
    title: "The majority of adults (25+) only..."
  }];
}, "LegendDiv");
//legend.startup();
vc //Applies the generated renderer to the feature layer
function applyRenderer(response){
  edLayer.setRenderer(response.renderer);
  edLayer.redraw();
  if(!edLayer.visible){
    edLayer.setVisibility(true);
  }
  legend.refresh();
}
```









U.S. Educational Attainment

Select the variables to include in the predominance map for exploring adult educational attainment by U.S. county.

All educational attainment

The majority of adults (25+) only ...

-  Did not finish high school
-  Graduated high school
-  Attended some college
-  Completed a bachelor's degree
-  Completed a post-graduate degree
-  Other

Strength of predominance

-  > 48
-  < 20

Q & A

Rate This Session

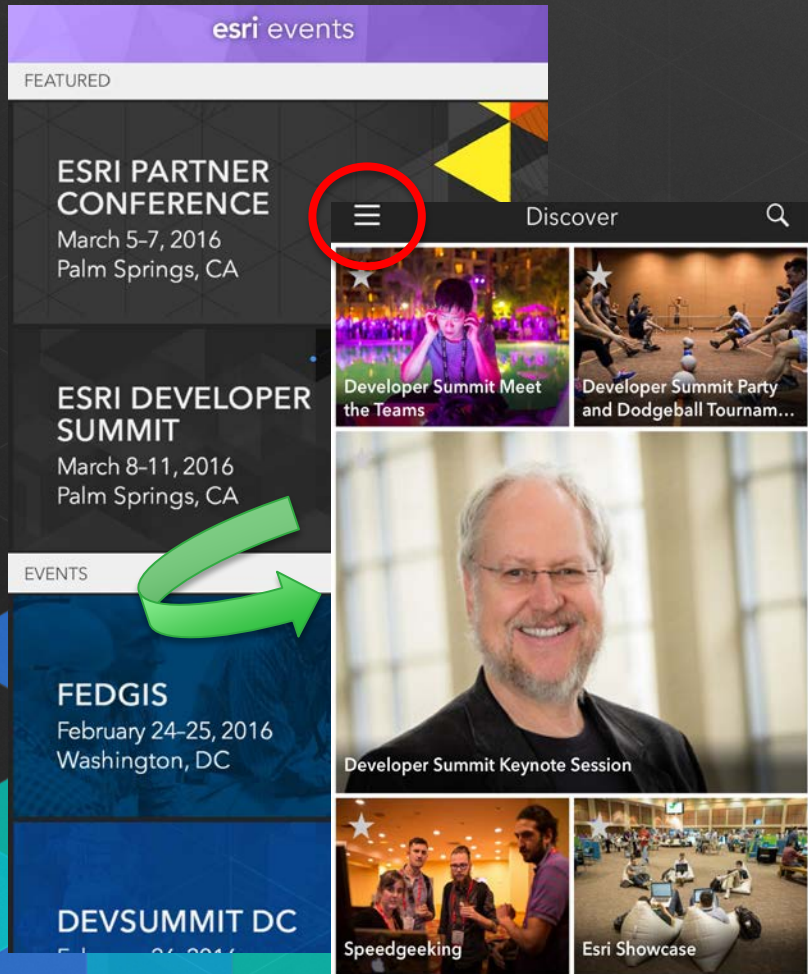
ArcGIS API for JavaScript: Data Visualization

Jeremy Bartley
Jim Herries @jherries

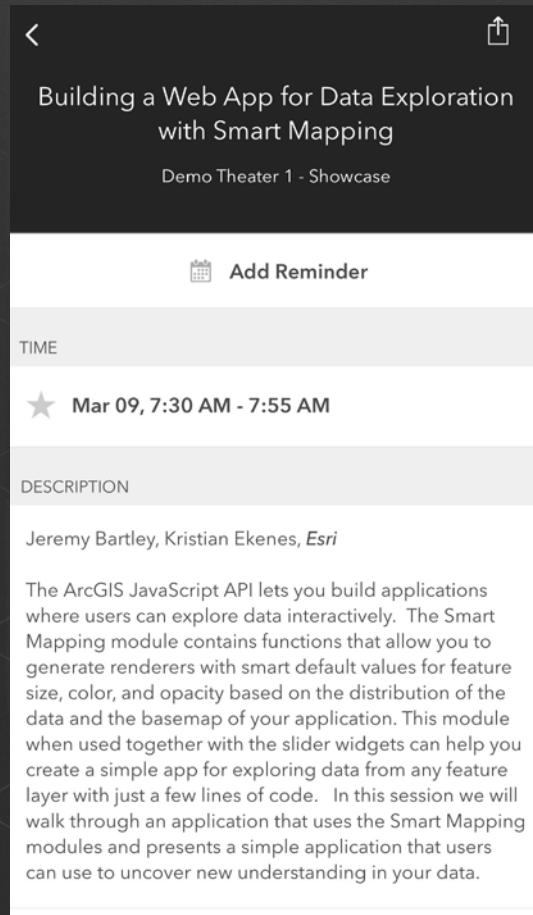


Please rate this Session

Download the Esri Events app and find your event

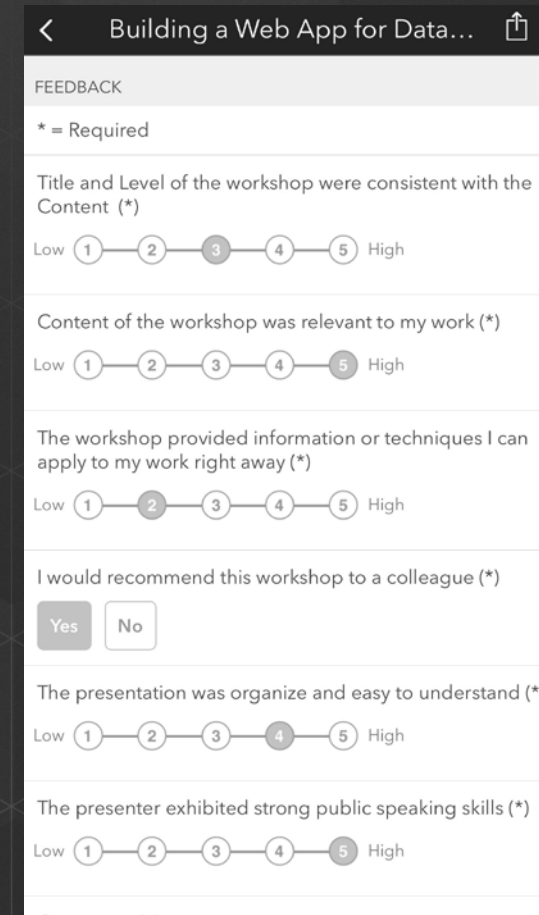


Select the session you attended



ArcGIS API for JavaScript: Data Visualization

Scroll down to the
“Feedback” section



Complete Answers,
add a Comment,
and Select “Submit”

