ArcGIS API for JavaScript: Amazing Mapping Apps

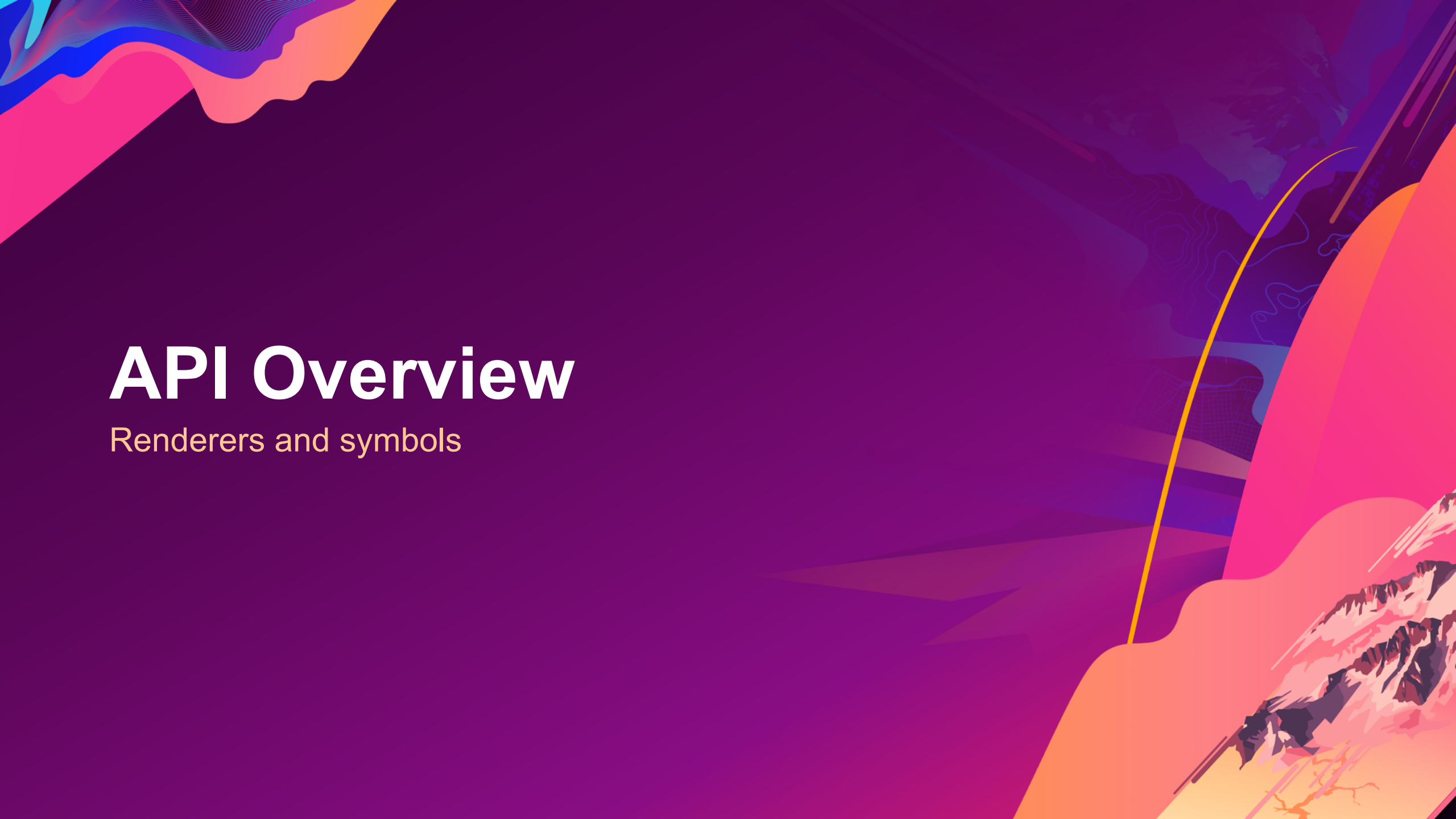
Anne Fitz, Jeremy Bartley

2021 ESRI USER CONFERENCE

Discover how to visualize geodata in meaningful ways using the ArcGIS API for JavaScript

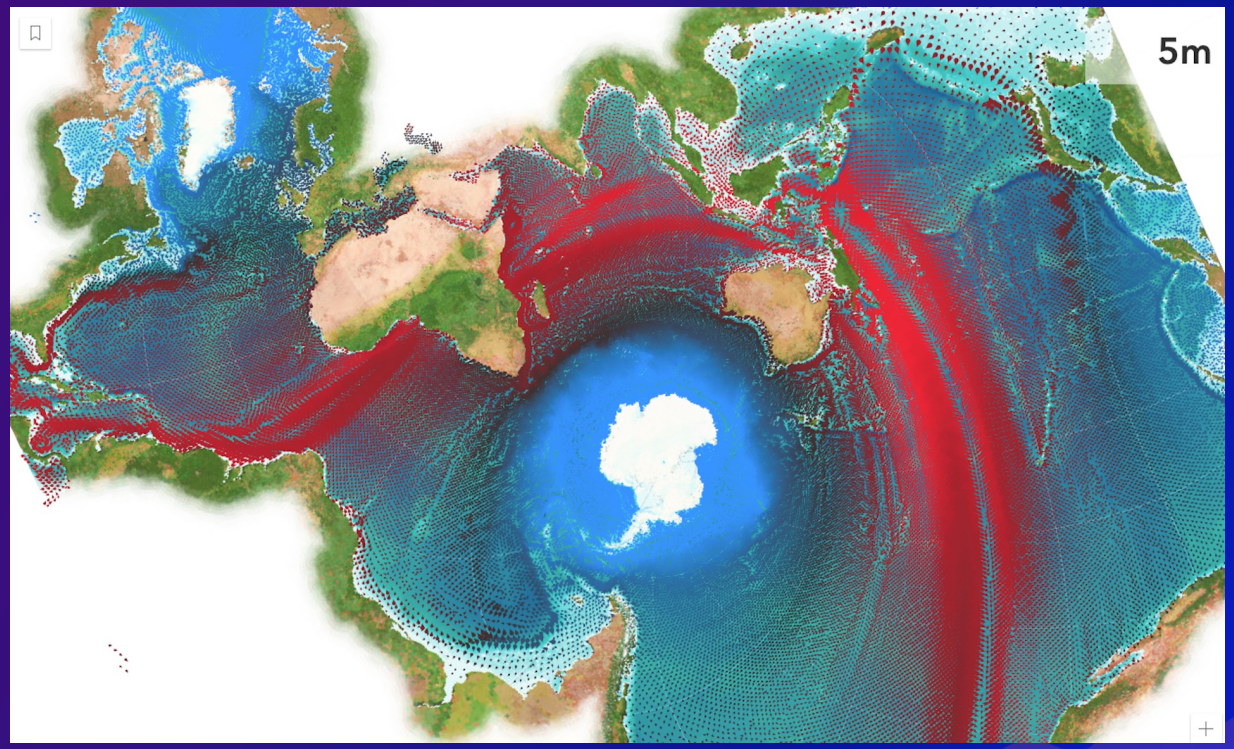
API Overview

Renderers and symbols



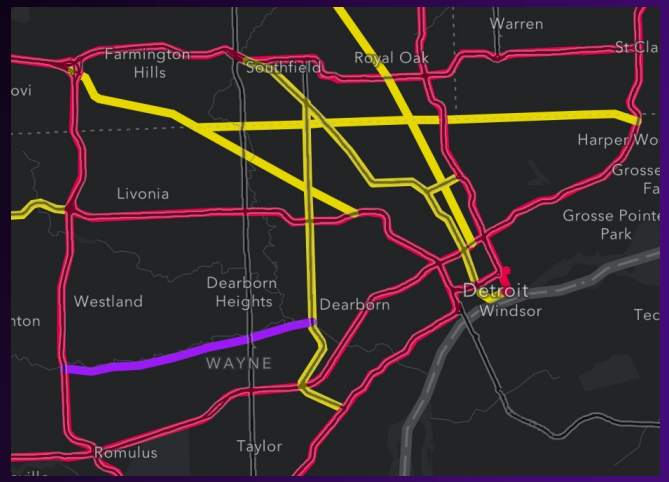
What can we visualize?

- Where?
- What?
- How much?
- When?
- Multivariate



Symbols

Symbol primitives



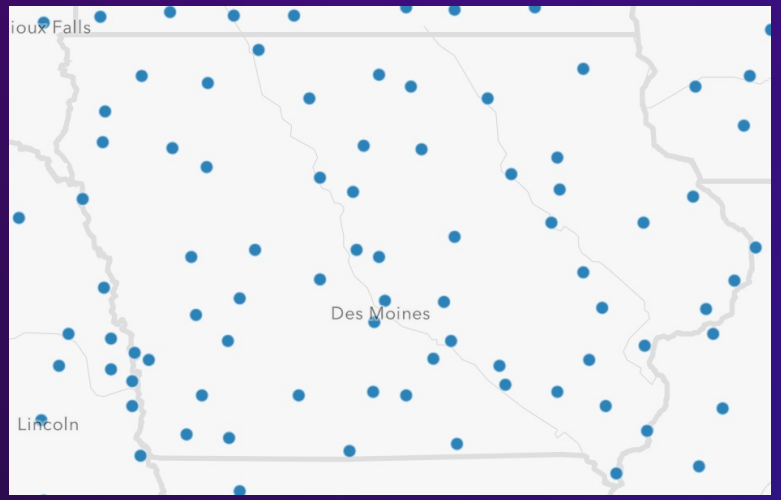
SimpleLineSymbol

```
const sls = {
  type: "simple-line",
  width: 1,
  color: [255, 255, 255, 1],
  style: "solid",
  cap: "round",
  join: "round"
}
```

SimpleFillSymbol

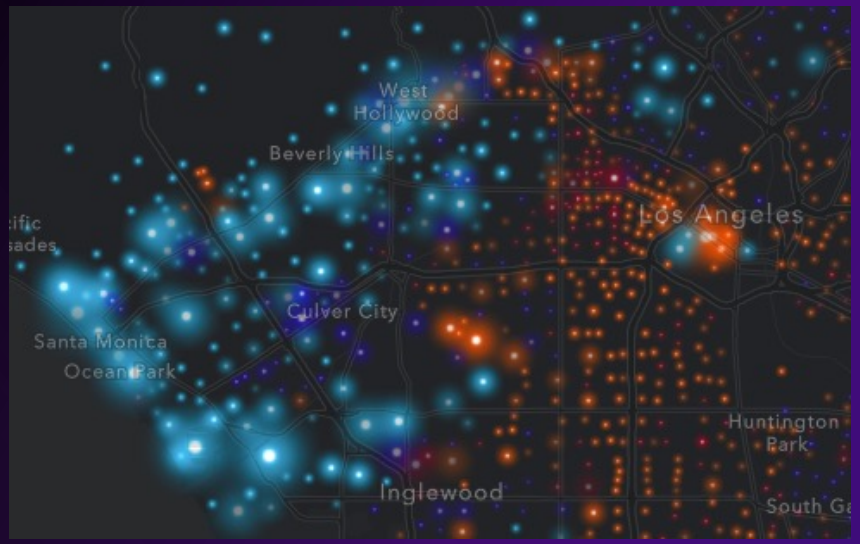
```
const sfs = {
  type: "simple-fill",
  color: [0, 0, 0, 0.25],
  style: "solid",
  outline: {
    width: 1,
    color: [255, 255, 255, 1]
  }
}
```

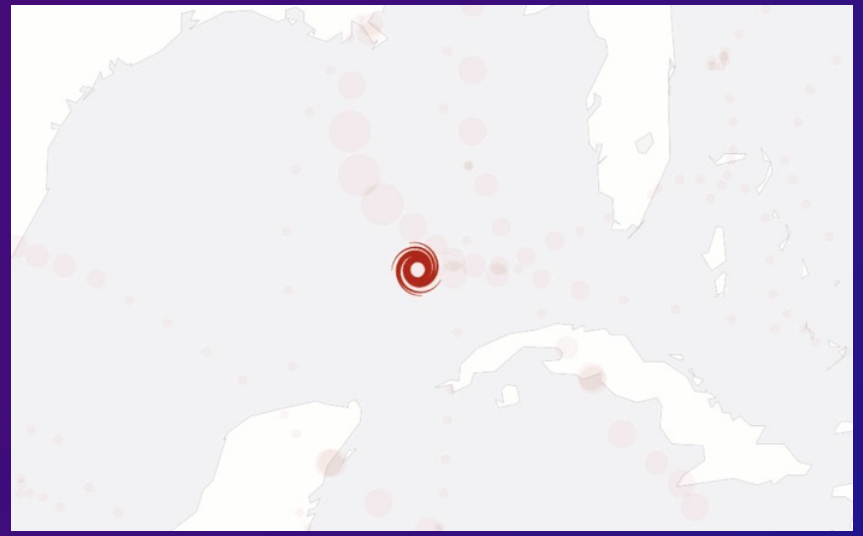
SimpleMarkerSymbol




```
const sms = {
  type: "simple-marker",
  color: [255, 255, 255, 0.25],
  size: 12,
  style: "circle",
  outline: {
    width: 1,
    color: [255, 255, 255, 1]
  }
}
```

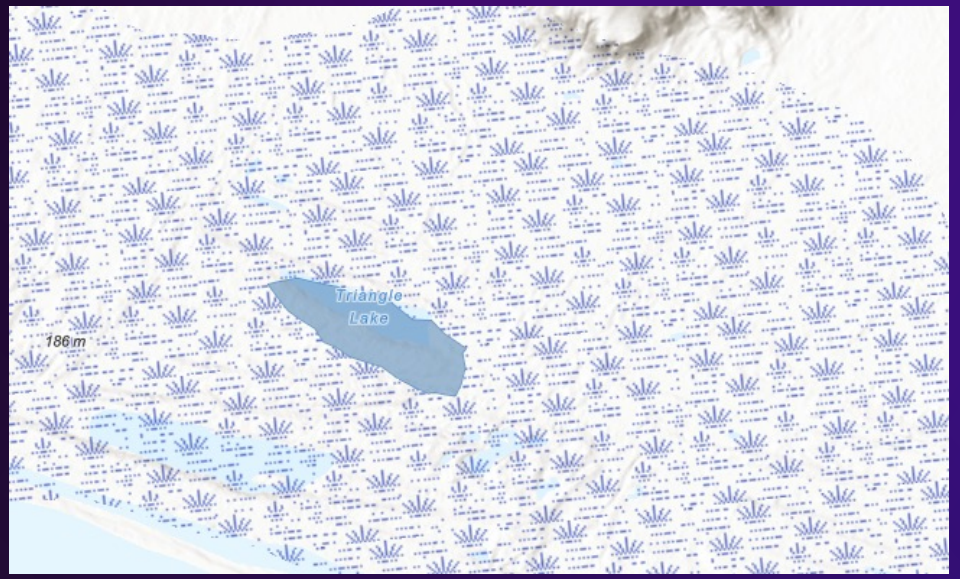
Picture symbols

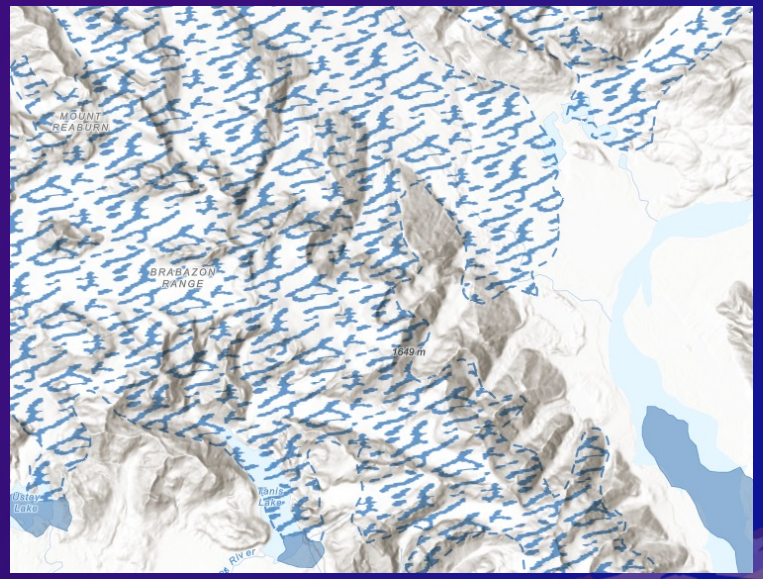




PictureMarkerSymbol

```
const pms = {
  type: "picture-marker",
  url: "image-url",
  height: 12,
  width: 12
}
```





PictureFillSymbol

```
const pfs = {
  type: "picture-fill",
  url: "swamp.png",
  width: 12,
  height: 12,
  xoffset: 0,
  yoffset: 0
}
```

CIMSymbol

High quality, scalable

Scaled vector symbol

Scaled image

Symbol layers

Primitive Overrides

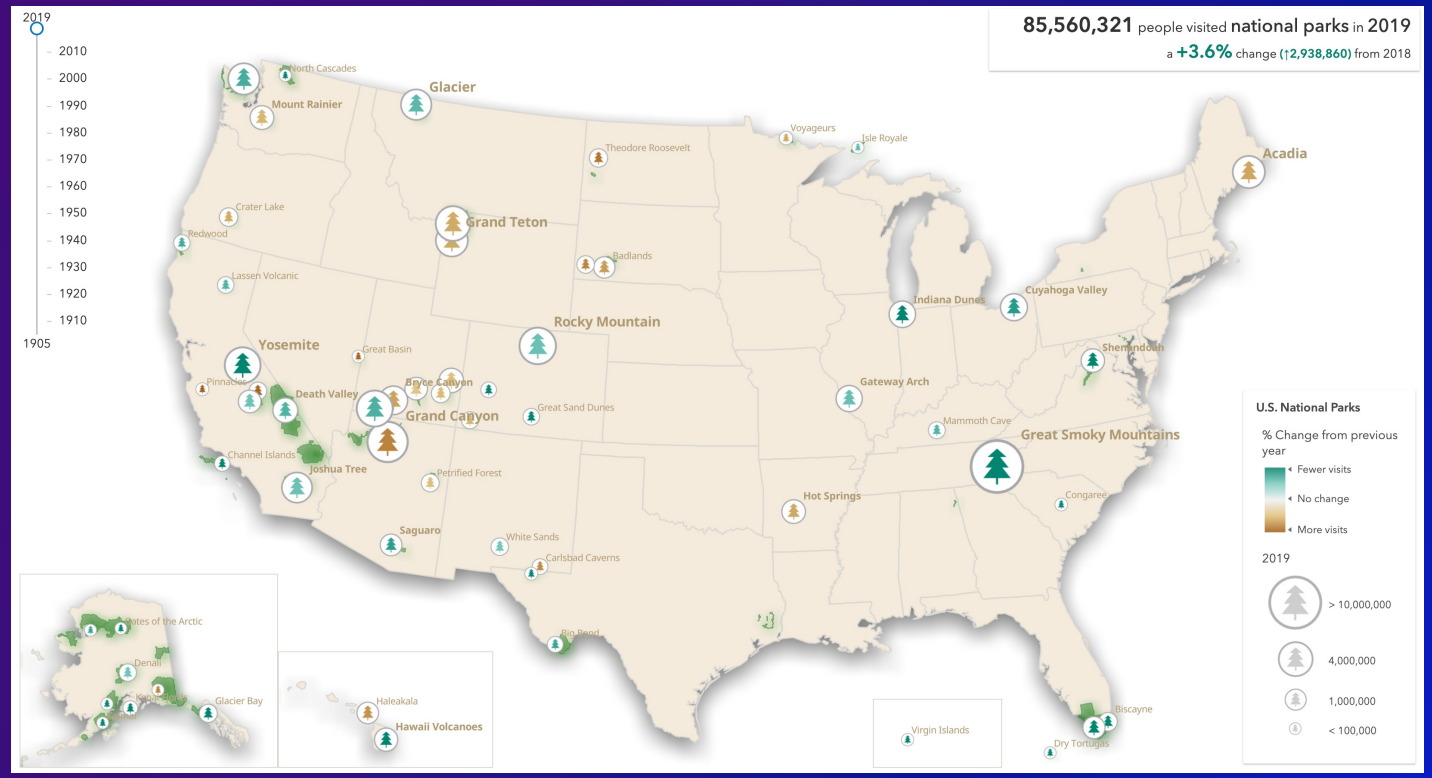
Dynamically update attributes of an individual symbol layer using Arcade

WebStyleSymbol

2d styles

Cartographic Information Model (CIM)

- Vectors
- Scalable
- Multi-layer
- Overrides

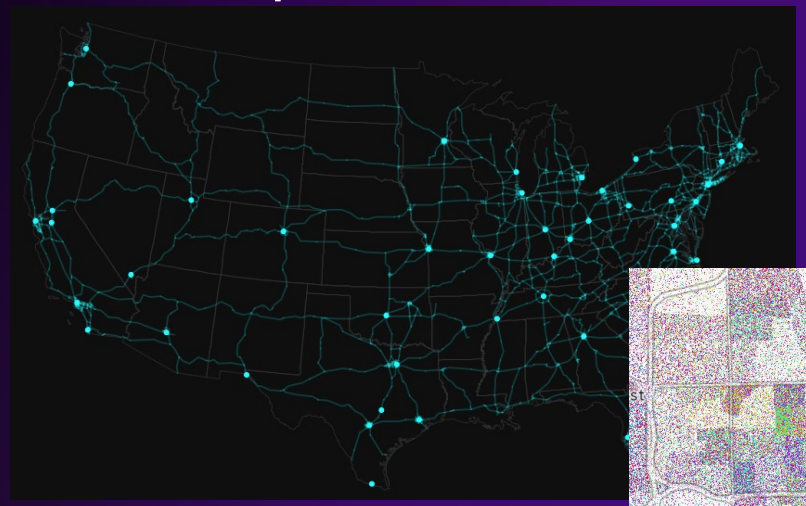



```
const webStyleSymbol = new WebStyleSymbol({
  name: "park",
  styleName: "Esri2DPointSymbolsStyle"
});
```

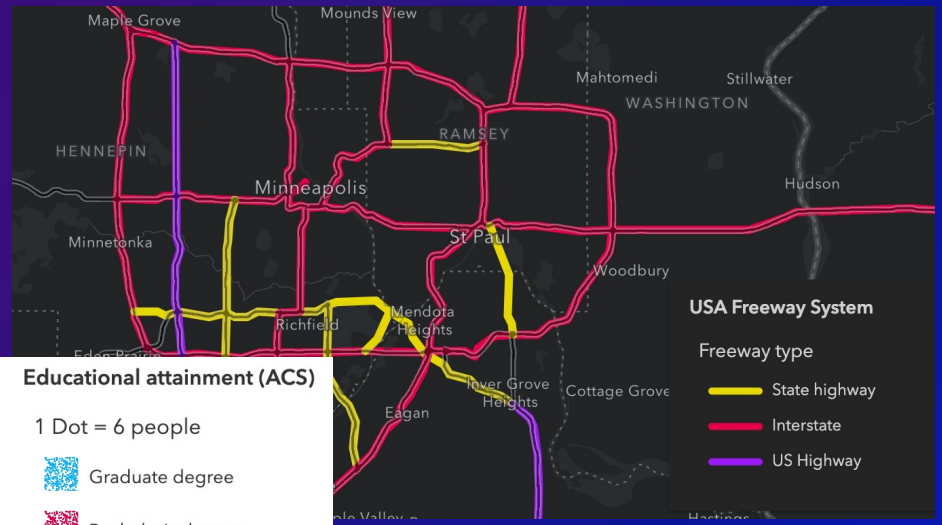
Renderers

Renderers

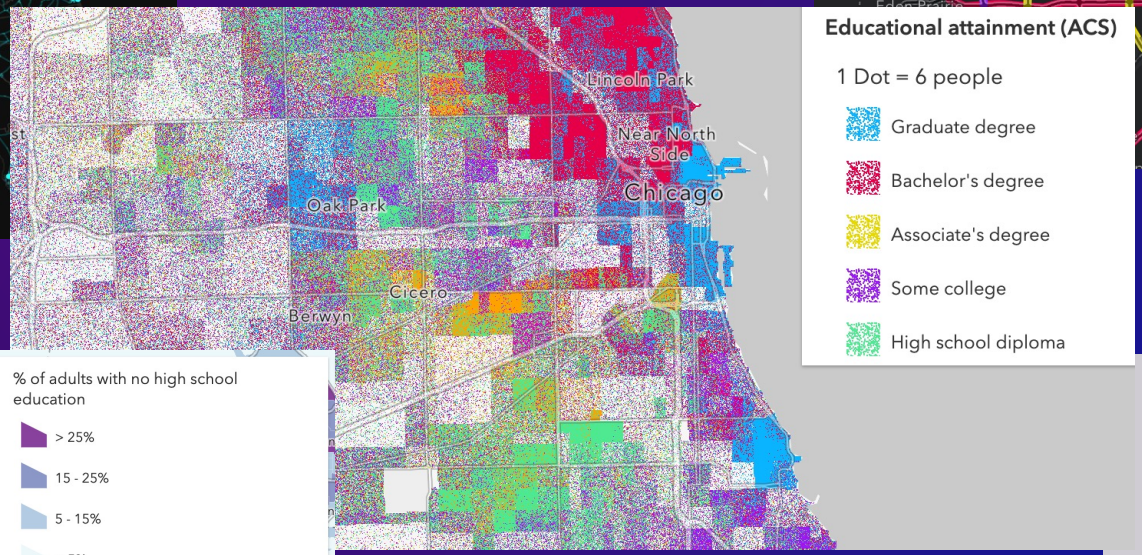
SimpleRenderer



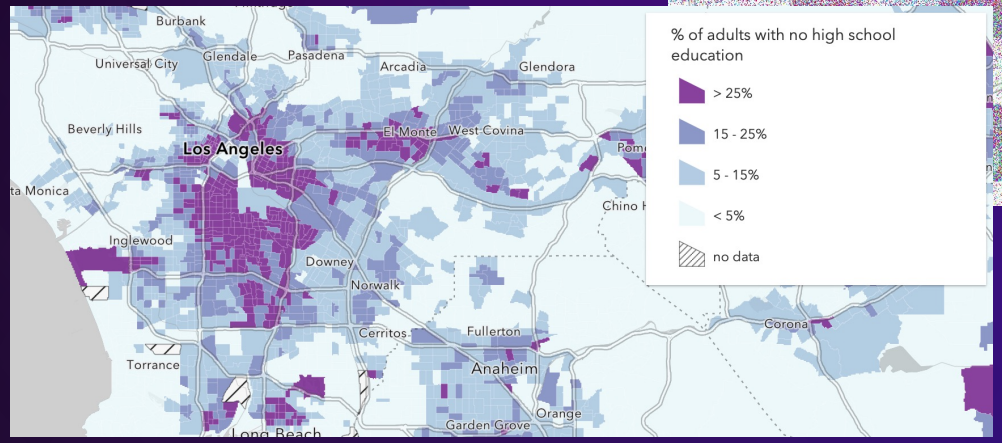
UniqueValueRenderer



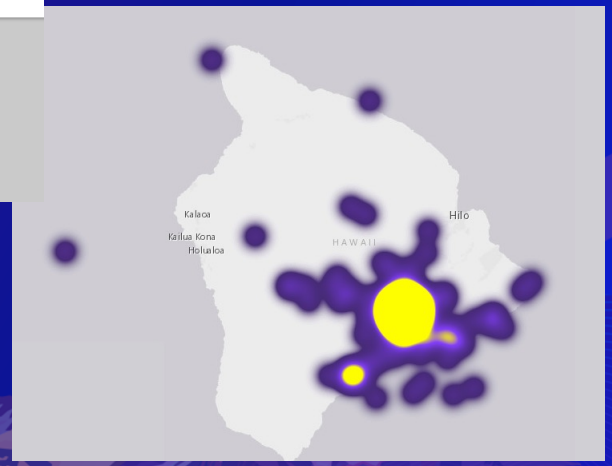
DotDensityRenderer



ClassBreaksRenderer



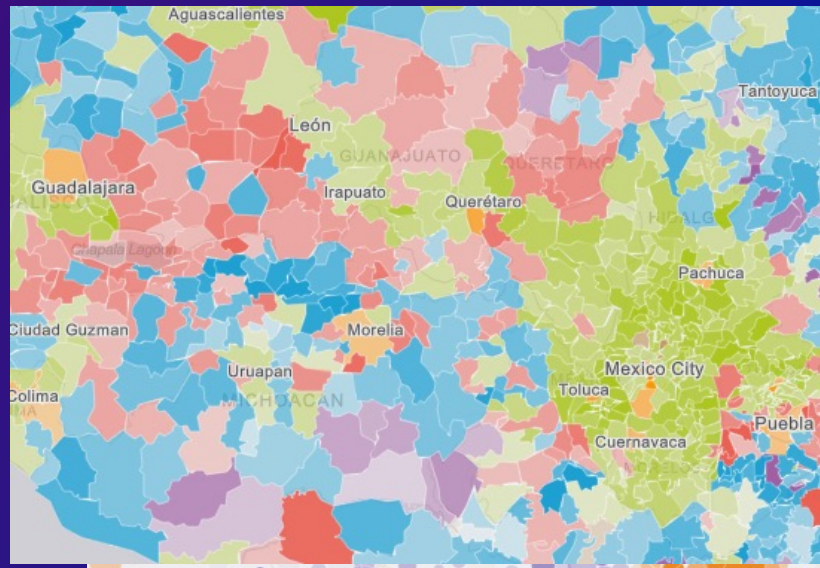
HeatmapRenderer

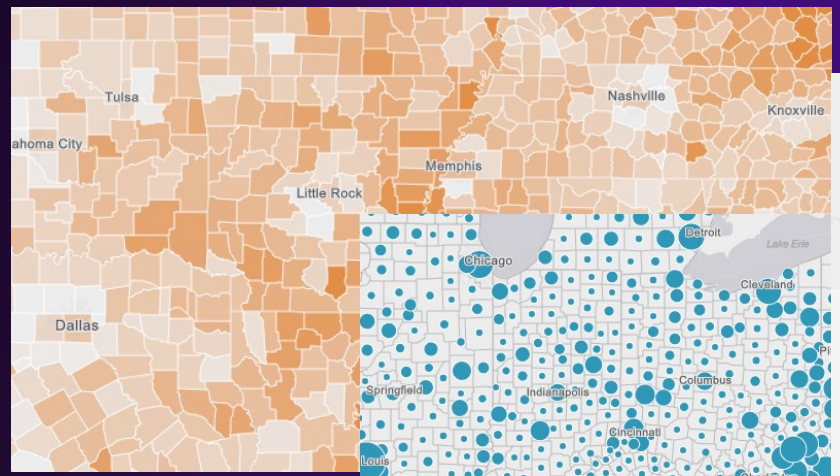


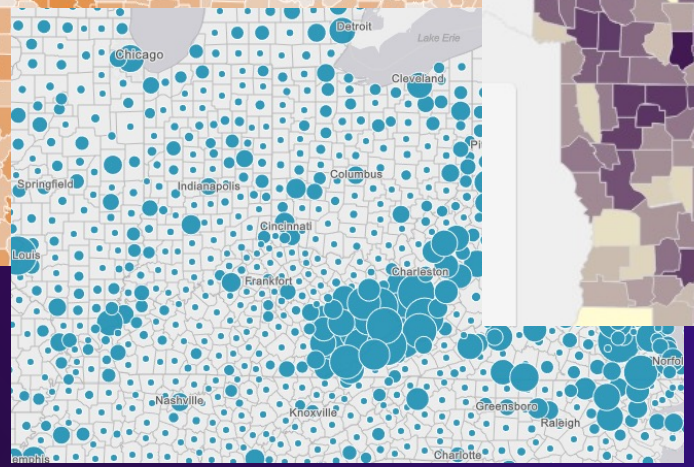
Visual variables

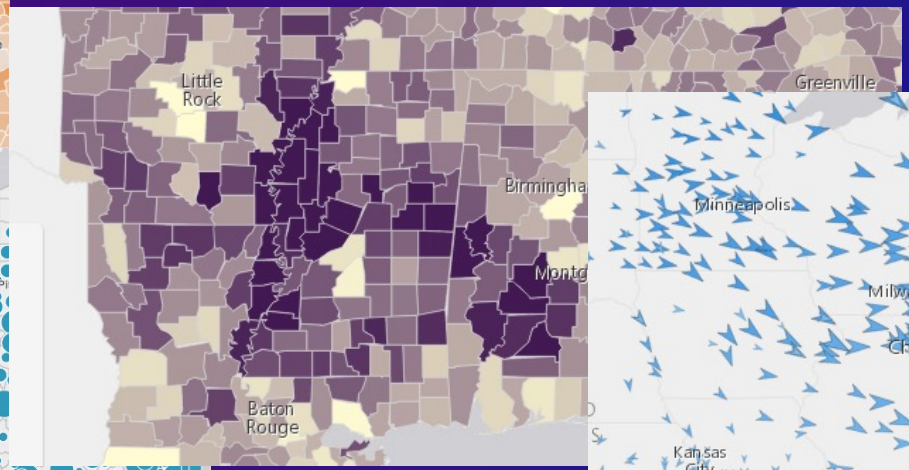
- Color
- Size
- Opacity
- Rotation

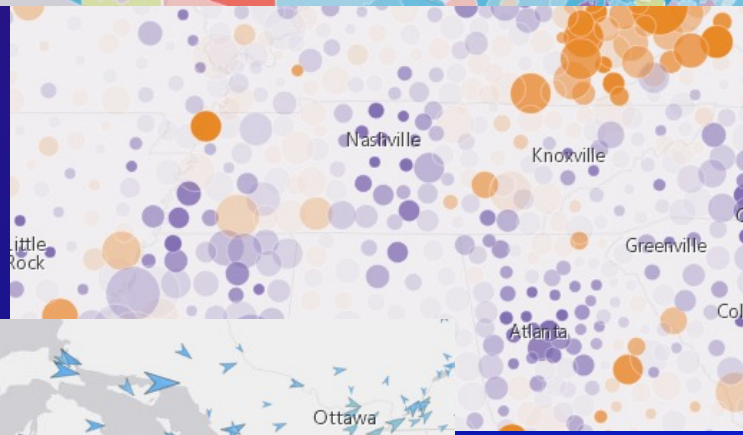
- A property of the renderer
- Overrides symbol properties with data
- For numeric data-driven continuous visualizations











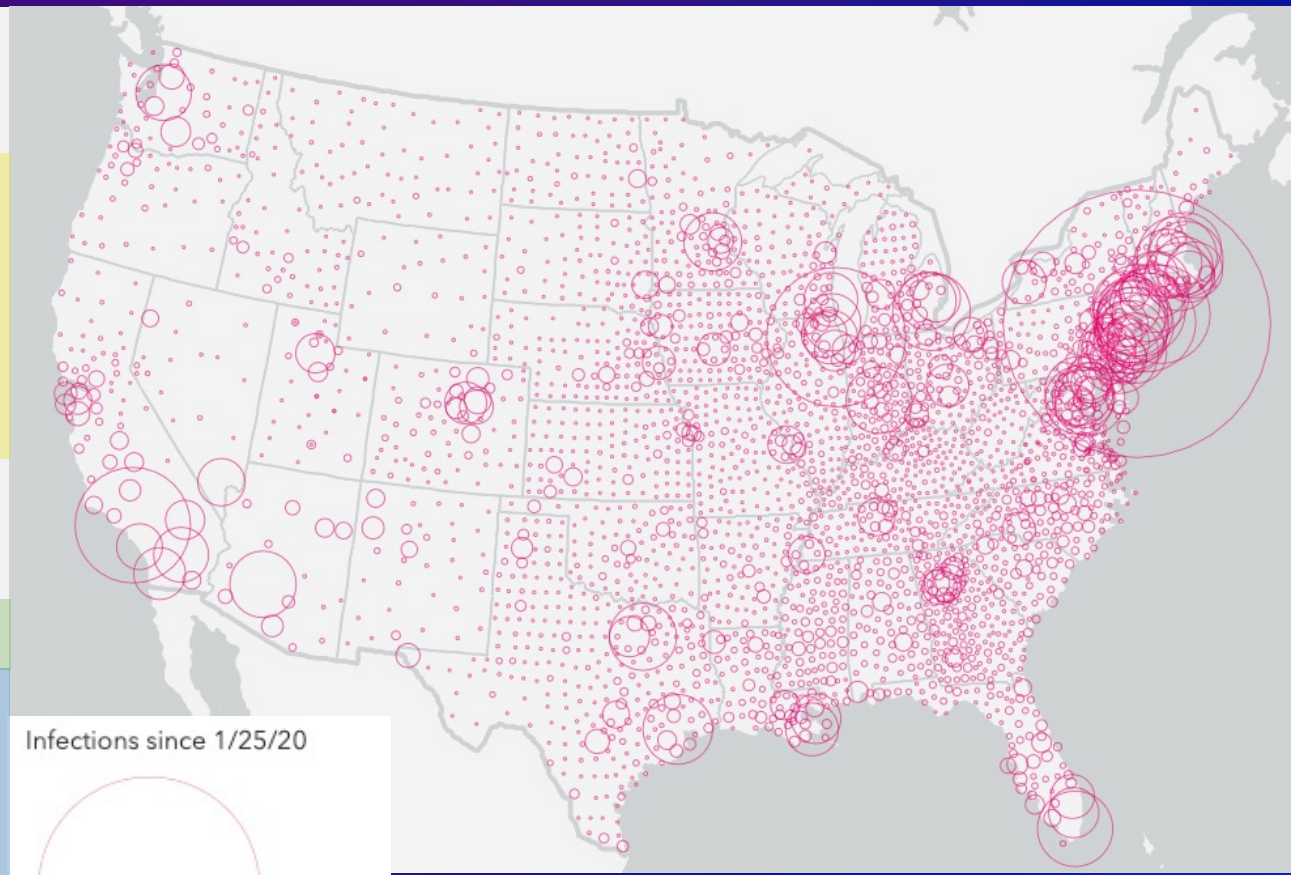
Data-driven visualization

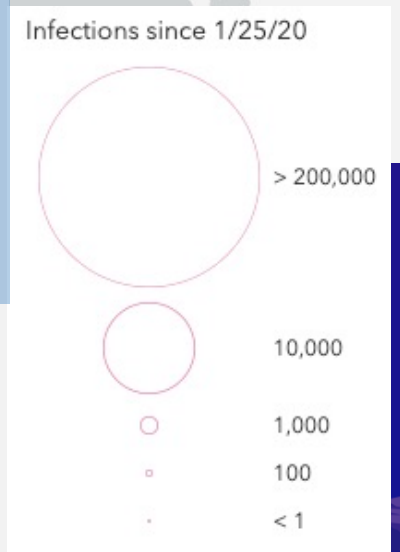
```
layer.renderer = new SimpleRenderer({
  symbol: new SimpleMarkerSymbol({
    style: "none",
    outline: new SimpleLineSymbol({
      color: new Color("rgba(227, 0, 106)"),
      width: 0.5
    })
  }),
  visualVariables: [
    new SizeVariable({
      field: "INFECTIONS_6_1_2020",
      stops: [
        { value: 1, size: "2px" },
        { value: 100, size: "4px" },
        { value: 1000, size: "10px" },
        { value: 10000, size: "50px" },
        { value: 200000, size: "200px" }
      ]
    })
  ]
});
```

Symbol

Data

Size driven by data





Field value

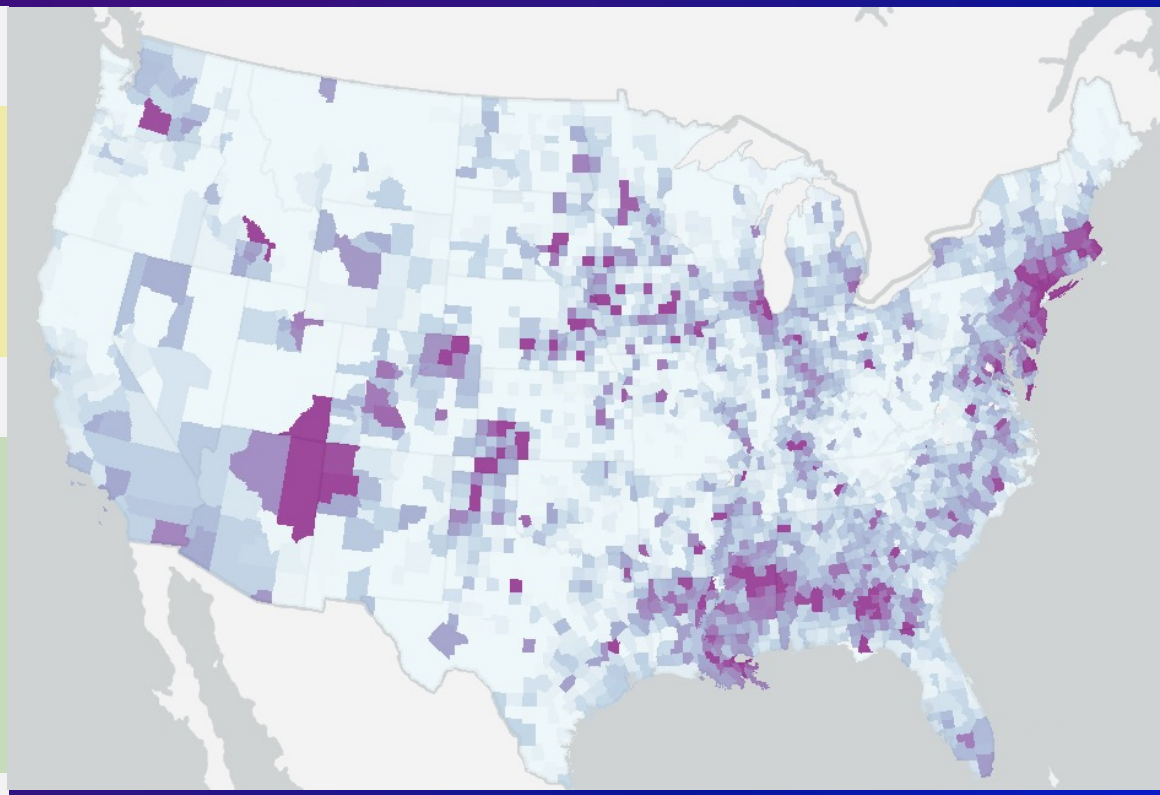
Data-driven visualization

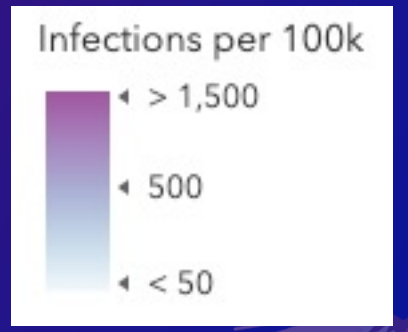
```
layer.renderer = new SimpleRenderer({
  symbol: new SimpleFillSymbol({
    outline: new SimpleLineSymbol({
      color: "rgba(128,128,128,0.4)",
      width: 0
    })
  }),
  visualVariables: [
    new ColorVariable({
      valueExpression: `
        var currentDayValue = $feature["DAYSTRING_05_31_2020"];
        var currentDaySplit = Split(currentDayValue, "|");
        var infections = Number(currentDaySplit[0]);
        var deaths = Number(currentDaySplit[1]);
        var population = $feature.Population_1;
        return (infections / population ) * 100000;
      `,
      valueExpressionTitle: `Infections per 100k`,
      stops: [
        { value: 50, color: "#edf8fb" },
        { value: 200, color: "#b3cde3" },
        { value: 500, color: "#8c96c6" },
        { value: 1000, color: "#8856a7" },
        { value: 1500, color: "#810f7c" }
      ]
    })
  ]
});
```

Symbol

Data

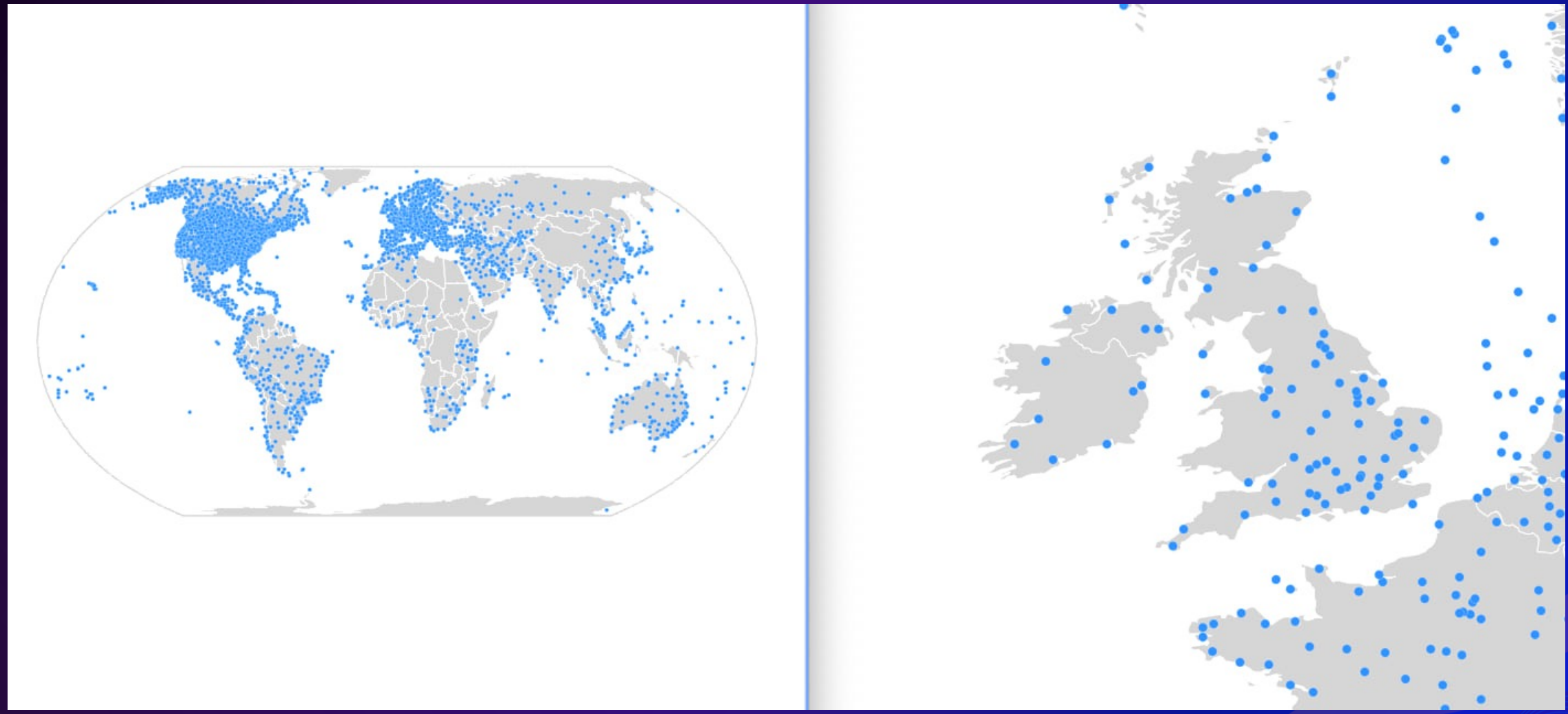
Color driven by data





Arcade expression

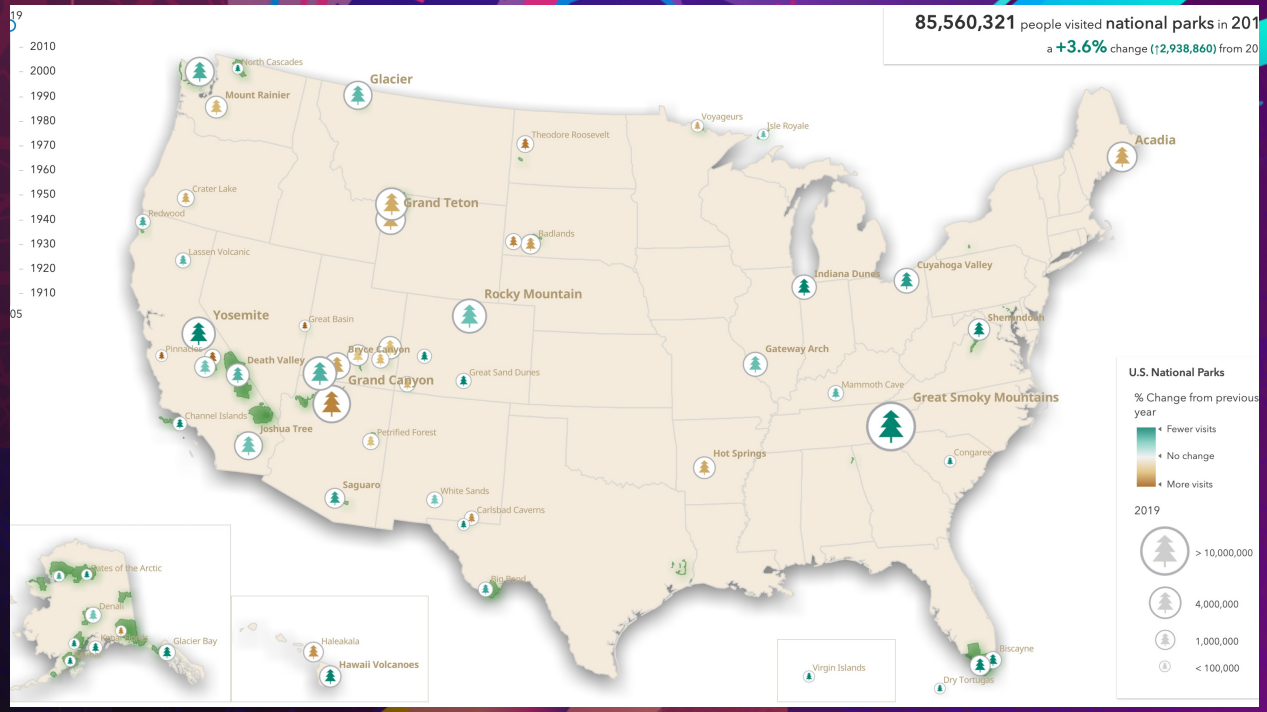
Symbol Size by Scale



ArcGIS blog: How and why to size symbols by scale in web maps

Symbol Size by Scale

```
renderer.visualVariables = [{
  type: "size",
  field: "velocity",
  minDataValue: 0,
  maxDataValue: 0.5,
  minSize: {
    type: "size",
    valueExpression: "$view.scale",
    stops: [
      { value: 7812500, size: "12px" },
      { value: 15625000, size: "8px" },
      { value: 31250000, size: "6px" },
      { value: 125000000, size: "2px" } ]
  },
  maxSize: {
    type: "size",
    valueExpression: "$view.scale",
    stops: [
      { value: 7812500, size: "32px" },
      { value: 15625000, size: "20px" },
      { value: 31250000, size: "16px" },
      { value: 125000000, size: "6px" }
    ]
  }
}];
```



National Park Explorer

Anne Fitz

Layer Blending

Layer blendMode

- Blend modes are used to blend layers together to create an interesting effect in a layer, or even to produce what seems like a new layer.
  - Top layer is a layer that has a blend mode applied. All layers underneath the top layer are background layers.

VintageLayer.blendMode = "color";

Blend modes categories

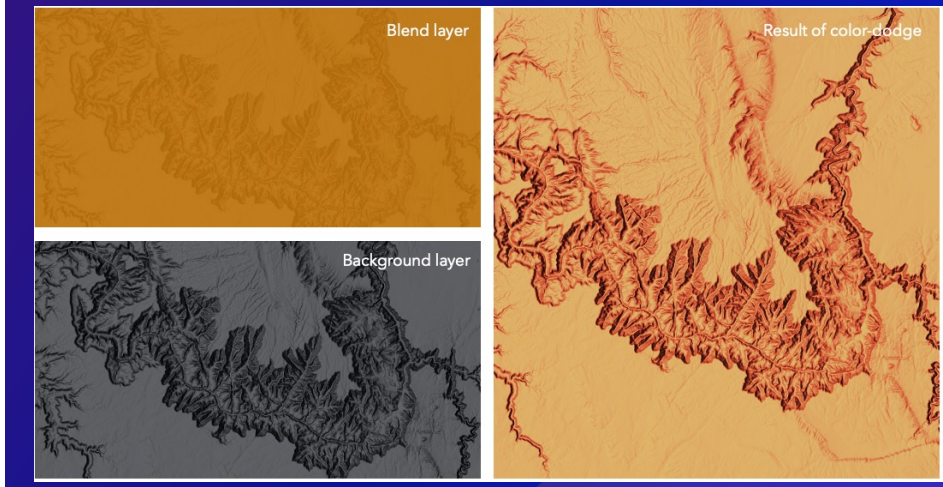
- The blendMode is a string property on most layers
- Default blend mode is normal
- Lighten blend modes
  - lighten, lighter, screen, plus, color-dodge
- Darken blend modes
  - darken, multiply, color-burn
- Contrast blend modes
  - overlay, soft-light, hard-light, vivid-light
- Component blend modes
  - hue, saturation, luminosity, color
- Composite blend modes
  - destination... and source...
- Invert blend modes
  - difference, exclusion, minus and invert

Lighten blend modes

- Some blend modes create lighter results than all layers.
  - Black becomes transparent, white stays white. Any color that is lighter than black will lighten.
  - Useful when lightening dark colors of the top/blend layer.

Blend mode	Description
`lighten`	Compares top and background layers and retains the lighter color in the top layer. Colors in the top layer become transparent if they are darker than the overlapping colors in the background layer allowing the background layer to show through completely. Can be thought of as the opposite of `darken` blend mode.
`lighter`	Colors in top and background layers are multiplied by their alphas (layer opacity and layer's data opacity. Then the resulting colors are added together. All overlapping midrange colors are lightened in the top layer. The opacity of layer and layer's data will affect the blend result.
`plus`	Colors in top and background layers are added together. All overlapping midrange colors are lightened in the top layer. This mode is also known as `add` or `linear-dodge`.
`screen`	Inverts colors in top and background, multiplied, and then inverted again. The resulting colors will be lighter than the original color with less contrast. Screen can produce many different levels of brightening depending on the luminosity values of the top layer. Can be thought of as the opposite of the `multiply` mode.
`color-dodge`	Creates a brighter effect by decreasing the contrast between the top and background layers, resulting in saturated mid-tones and bright highlights.

topLayer.blendMode= "color-dodge"

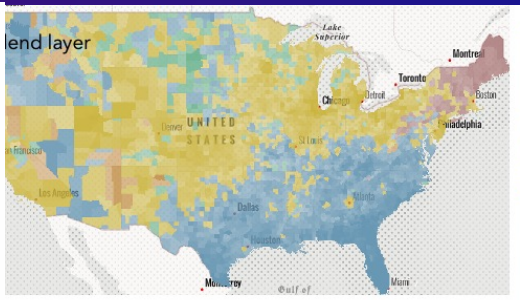


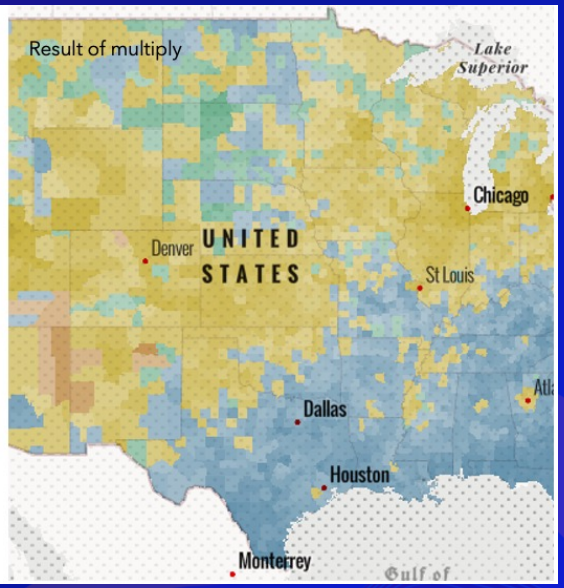
Darken blend modes

- Some blend modes create darker results than all layers.
  - White becomes transparent, black stays black. Any color that is darker than white is going to darken.
  - Can be used to highlight shadows, labels, show contrast, or accentuate an aspect of a map.

topLayer.blendMode = "multiply";

Blend mode	Description
`darken`	Emphasizes the darkest parts of overlapping layers. Colors in the top layer become transparent if they are lighter than the overlapping colors in the background layer, allowing the background layer to show through completely.
`multiply`	Emphasizes the darkest parts of overlapping layers by multiplying colors of the top layer and the background layer. Midrange colors from top and background layers are mixed together more evenly.
`color-burn`	Intensifies the dark areas in all layers. It increases the contrast between top and background layers, by tinting colors in overlapping area towards the top color. To do this it inverts colors of the background layer, divides the result by colors of the top layer, then inverts the results.

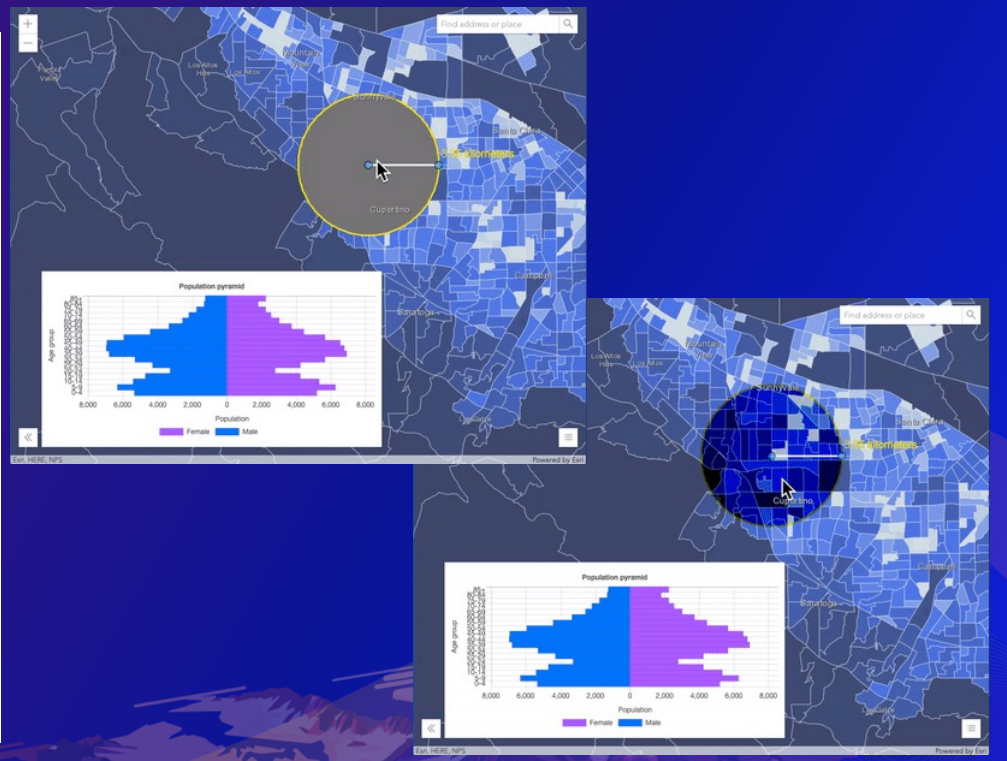




Contrast blend modes

- Some blend modes create contrast by using lightening or darkening blend modes to create the blend
  - Lightens the colors lighter than 50% gray, and darkens the colors darker than 50% gray. 50% gray will be transparent.
  - Can be used to increase the contrast and saturation to have more vibrant colors and give a punch to your layers.

Blend mode	Description
`overlay`	Uses a combination of `multiply` and `screen` modes to darken and lighten colors in the top layer with the background layer always shining through. The result is darker color values in the background layer intensify the top layer, while lighter colors in the background layer wash out overlapping areas in the top layer.
`soft-light`	Applies a half strength `screen` mode to lighter areas and and half strength `multiply` mode to darken areas of the top layer. You can think of the `soft-light` as a softer version of the `overlay` mode.
`hard-light`	Multiplies or screens the colors, depending on colors of the top layer. The effect is similar to shining a harsh spotlight on the top layer.
`vivid-light`	Uses a combination of `color-burn` or `color-dodge` by increasing or decreasing the contrast, depending on colors in the top layer.

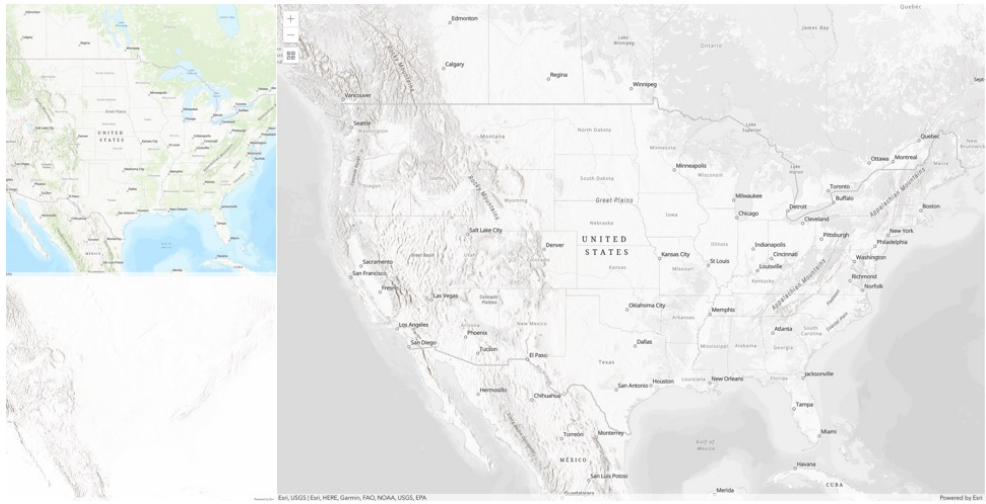


Component blend modes

- Some blend modes use primary color components: hue, saturation and luminosity to blend layers.
  - Use a feature layer with a simple renderer over any layer and set hue, saturation, color, or luminosity blend mode on thel ayer. With this technique, you create a brand new looking map.

Blend mode	Description
`hue`	Creates an effect with the hue of the top layer and the luminosity and saturation of the background layer.
`saturation`	Creates an effect with the saturation of the top layer and the hue and luminosity of the background layer. 50% gray with no saturation in the background layer will not produce any change.
`luminosity`	Creates effect with the luminosity of the top layer and the hue and saturation of the background layer. Can be thought of as the opposite of `color` blend mode.
`color`	Creates an effect with the hue and saturation of the top layer and the luminosity of the background layer. Can be thought of as the opposite of `luminosity` blend mode.

```
topLayer.blendMode =  "luminosity";
```

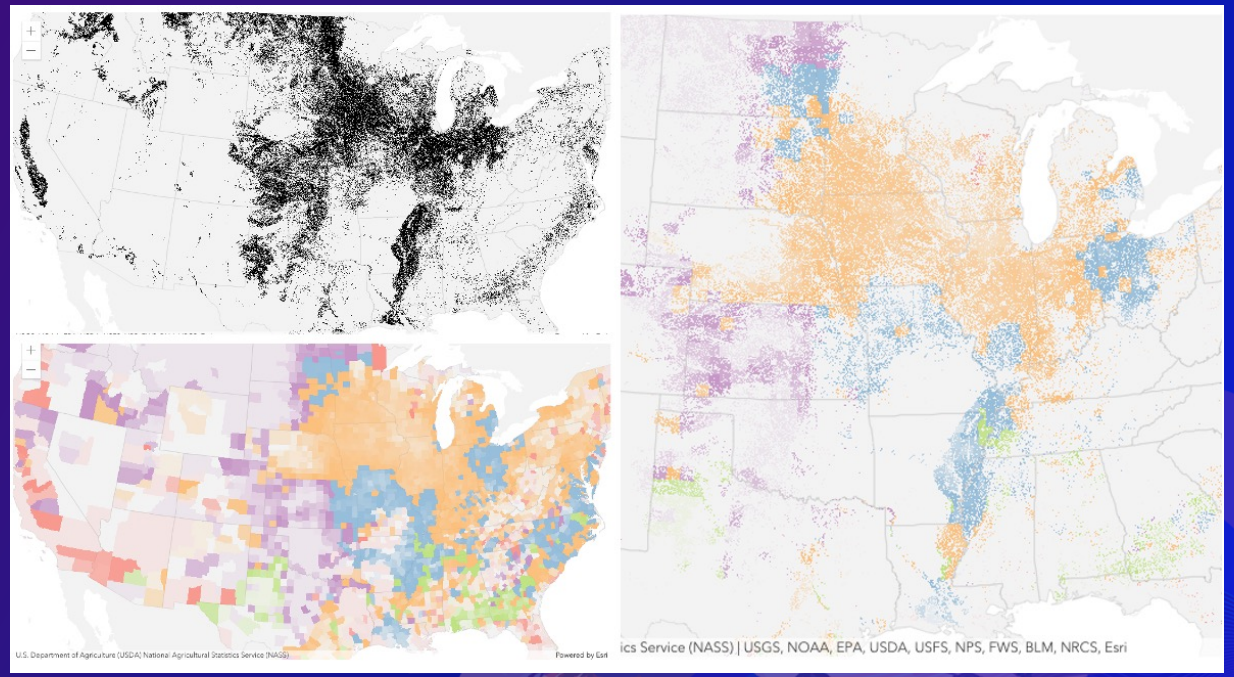


Composite blend modes

- Some blend modes can be used to mask the contents of top, background or both layers.
  - Destination modes are used to mask the data of the top layer with the data of the background layer.
  - Source modes are used to mask the data of the background layer with the data of the top layer.

Blend mode	Description
`destination-over`	Destination/background layer covers the top layer. The top layer is drawn underneath the destination layer. You'll see the top layer peek through wherever the background layer is transparent or has no data.
`destination-atop`	Destination/background layer is drawn only where it overlaps the top layer. The top layer is drawn underneath the background layer. You'll see the top layer peek through wherever the background layer is transparent or has no data.
`destination-in`	Destination/background layer is drawn only where it overlaps with the top layer. Everything else is made transparent.
`destination-out`	Destination/background layer is drawn where it doesn't overlap the top layer. Everything else is made transparent.
`source-atop`	Source/top layer is drawn only where it overlaps the background layer. You will see the background layer peek through where the source layer is transparent or has no data.
`source-in`	Source/top layer is drawn only where it overlaps with the background layer. Everything else is made transparent.
`source-out`	Source/top layer is drawn where it doesn't overlap the background layer. Everything else is made transparent.
`xor`	Top and background layers are made transparent where they overlap. Both layers are drawn normal everywhere else.

topLayer.blendMode = "destination-in";

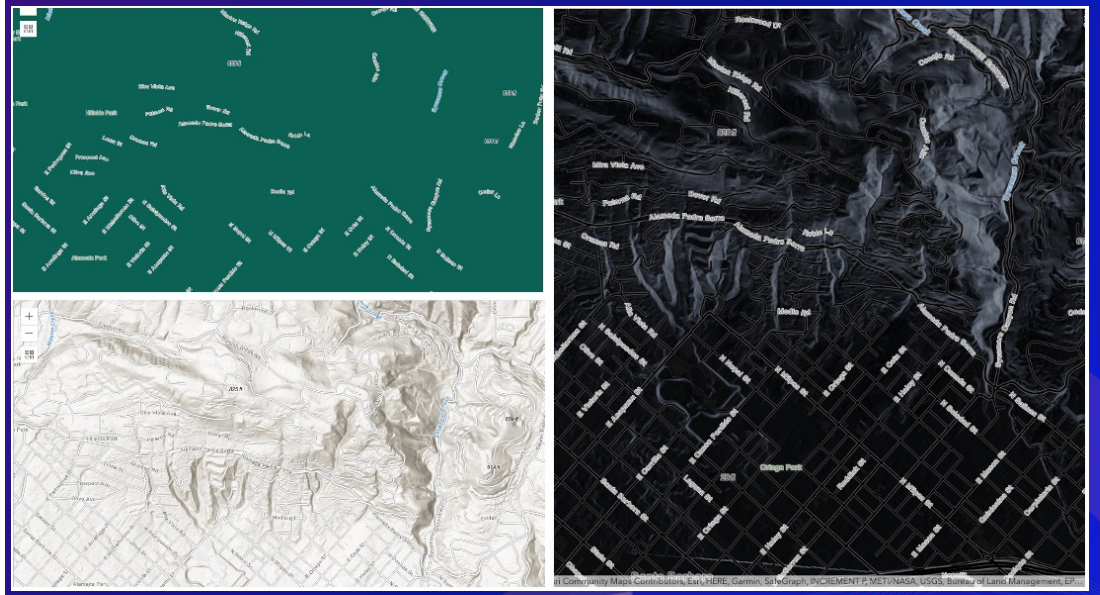


Invert blend modes

- Some blend modes either invert or cancel out colors depending on colors of the background layer.
  - Use difference or exclusion blend modes on two imagery layers of forest covers to visualize how forest covers changed from one year to another.
  - The invert blend mode can be used to turn any light basemap into a dark basemap

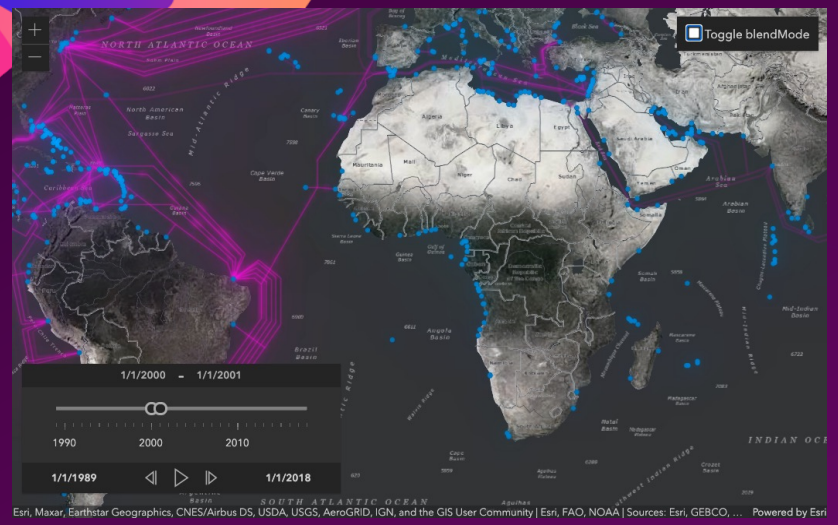
Blend mode	Description
`difference`	Subtracts the darker of the overlapping colors from the lighter color. When two pixels with the same value are subtracted, the result is black. Blending with black produces no change. Blending with white inverts the colors. This blending mode is useful for aligning layers with similar content.
`exclusion`	Similar to the `difference` blend mode, except that the resulting image is lighter overall. Overlapping areas with lighter color values are lightened, while darker overlapping color values become transparent.
`minus`	Subtracts colors of the top layer from colors of the background layer making the blend result darker. In the case of negative values, black is displayed.
`invert`	Inverts the background colors wherever the top and background layers overlap. The invert blend mode inverts the layer similar to a photographic negative.
`reflect`	This blend mode creates effects as if you added shiny objects or areas of light in the layer. Black pixels in the background layer are ignored as if they were transparent.

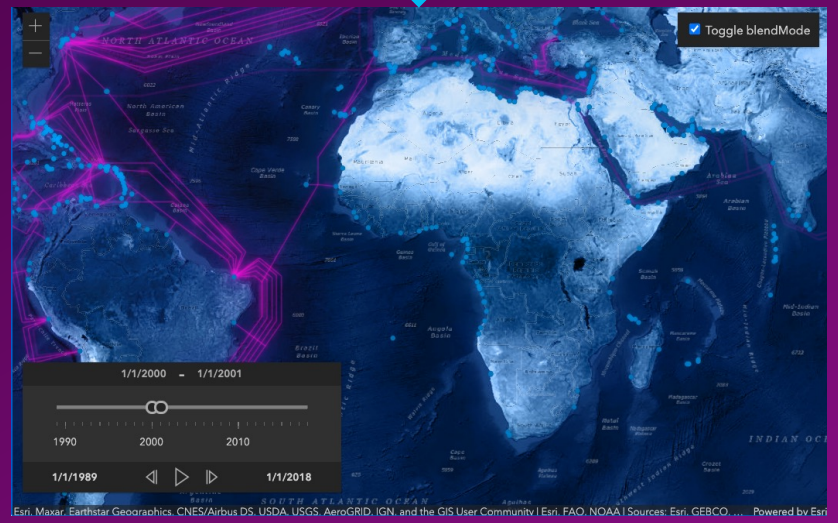
```
topLayer.blendMode =  "invert";
```



Layer blendMode - tips

- Order of all layers
- Layer opacity
- Opacity of features in layers
- Visibility of layers
- By default, the very bottom layer in a map is drawn on a transparent background. You can change the MapView's background color.





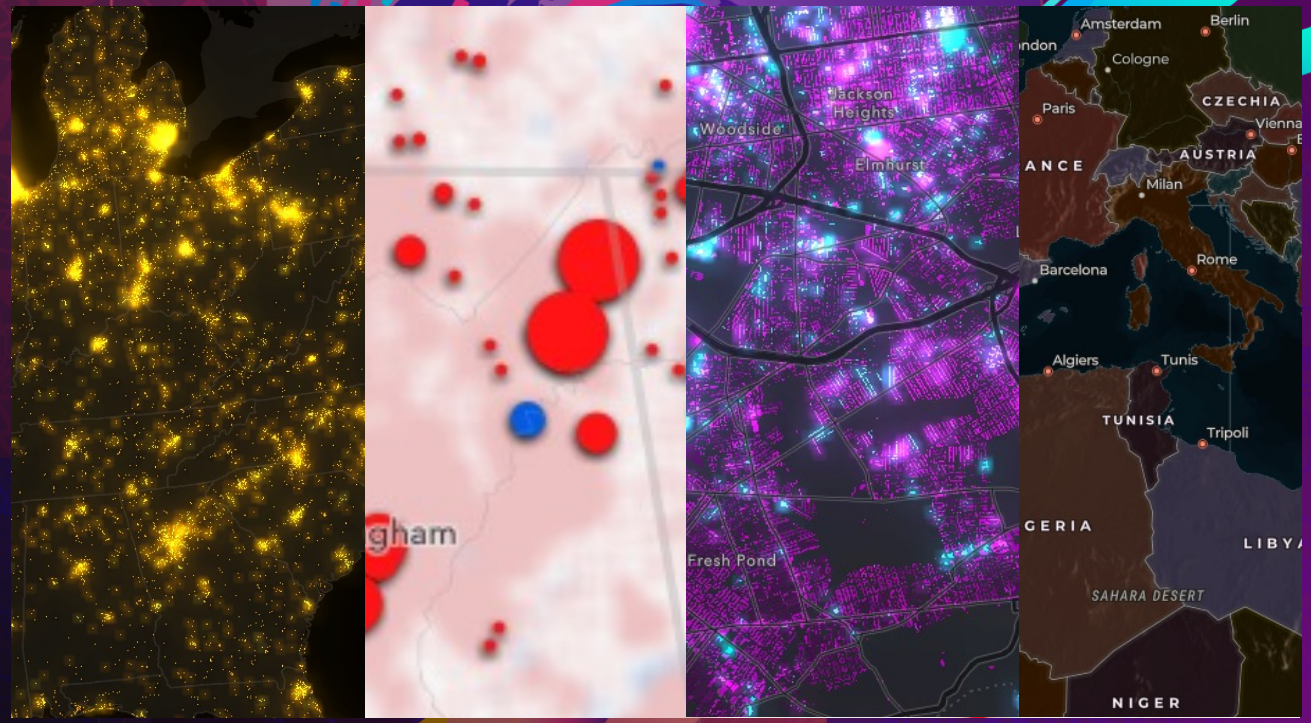
Submarine cables

Demo and source code

Map of Romania

Source code

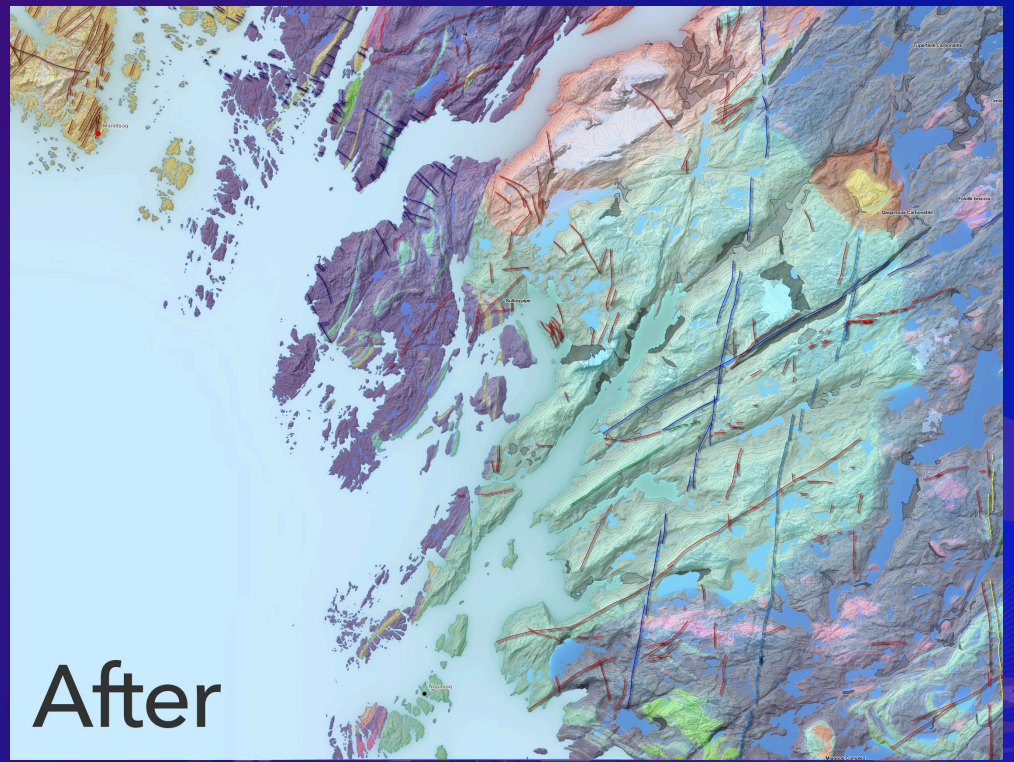
Live app



Layer and Feature Effects

Layer effects

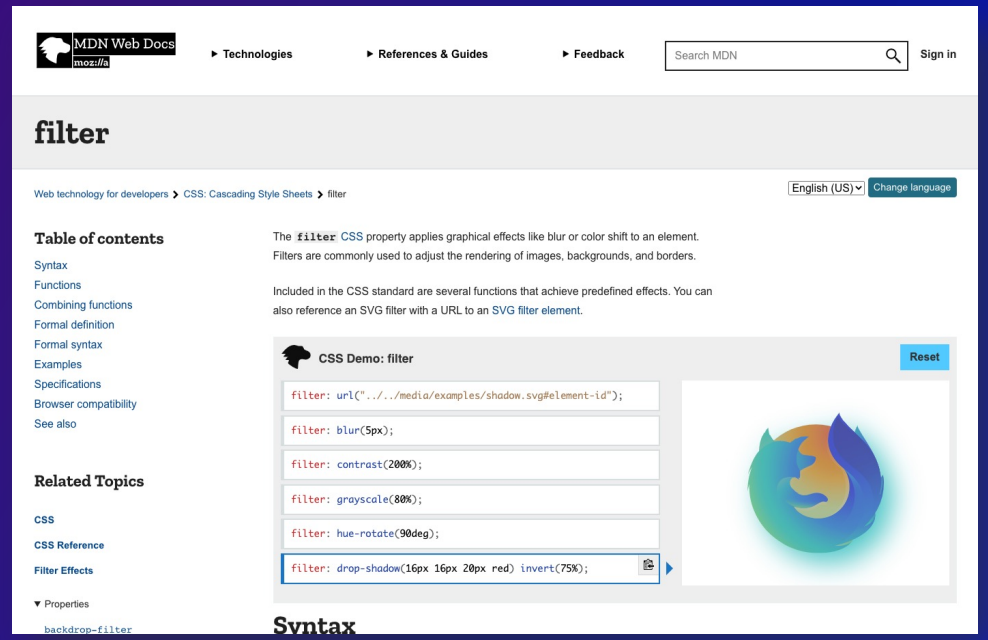
- Various filter functions that can be performed on a layer to achieve different visual effects like in a photo editing apps (Photoshop, GIMP).
- Available on all layers supported in 2D.



Layer effect

Syntax

- An effect is defined like CSS filter
  - If you know about CSS filters you are already proficient with layer filters
- Format
  - A string that describes a list of filter functions with parameters
  - Parameter units: percentage (ratio), lengths, color
- Available filter functions:
  - `blur, brightness, contrast, drop-shadow, grayscale, hue-rotate, invert, opacity, saturate, sepia`
  - `bloom`




```
layer.effect = "contrast(200%) blur(5px) drop-shadow(1px 2px 5px #000000)";
```

Layer effect

Scale dependency

- Effects are interpolated when user zooms
- Format
  - An array of objects with a scale and an effect value
  - Effect list must be compatible between scales

```
layer.effect = [
  { scale: 80000, value: 'opacity(50%)' },
  { scale: 20000, value: 'bloom(1 0px 65%)' }
];
```

Interpolation not possible
between opacity and bloom

```
layer.effect = [
  { scale: 80000, value: 'bloom(0 0px 65%) opacity(50%)' },
  { scale: 20000, value: 'bloom(1 0px 65%) opacity(100%)' }
];
```

```
layer.effect = [
  { scale: 80000, value: 'bloom(0 0px 65%) opacity(50%)' },
  { scale: 20000, value: 'bloom(1 0px 65%)' }
];
```

Effects Explorer

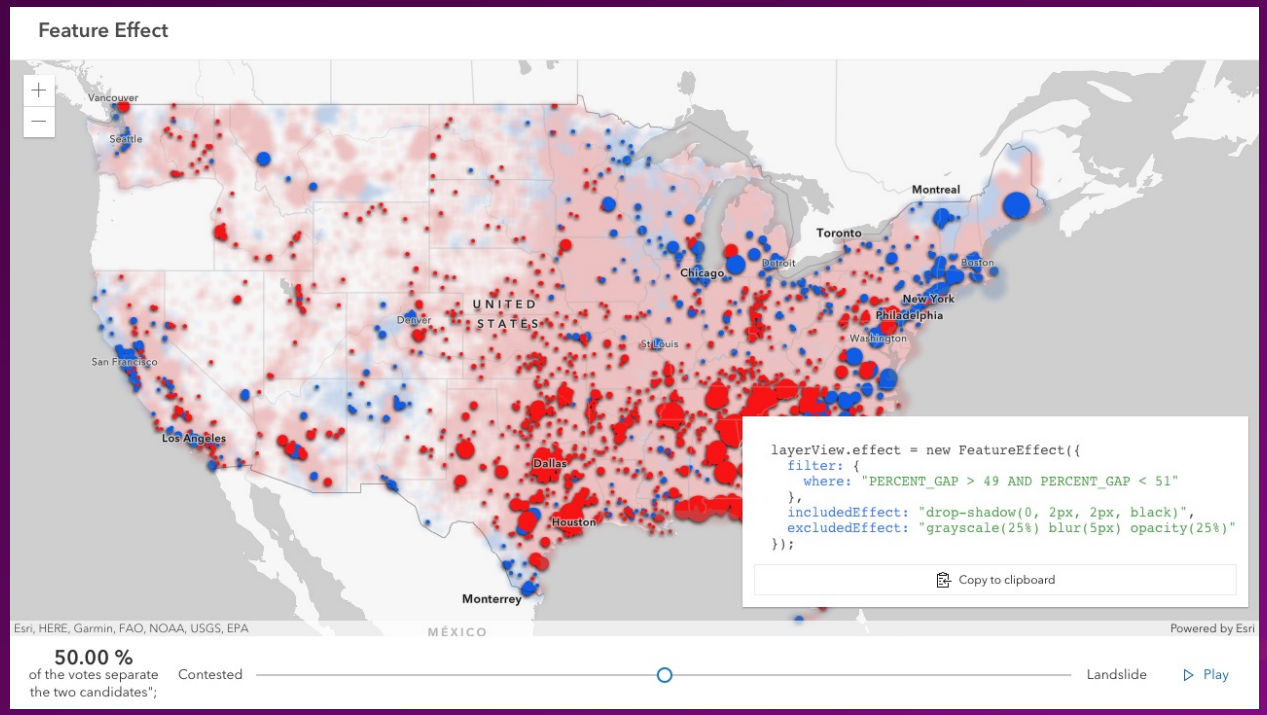
[Demo]

[Source code]

Feature Effects

- The effect allows for the selection of features via a `filter`, and an `includedEffect` and `excludedEffect` are applied to those features that respectively pass or fail the filter requirements.
- Available on feature layerviews in 2D: FeatureLayerView, GeoJSONLayerView, CSVLayerView, StreamLayerView

```
layerView.effect = new FeatureEffect({
  filter: {
    where: "PERCENT_GAP > 49 AND PERCENT_GAP < 51"
  },
  includedEffect: "drop-shadow(0, 2px, 2px, black)",
  excludedEffect: "grayscale(25%) blur(5px) opacity(25%)"
});
```



Feature Effects

Demo1 & Demo 2

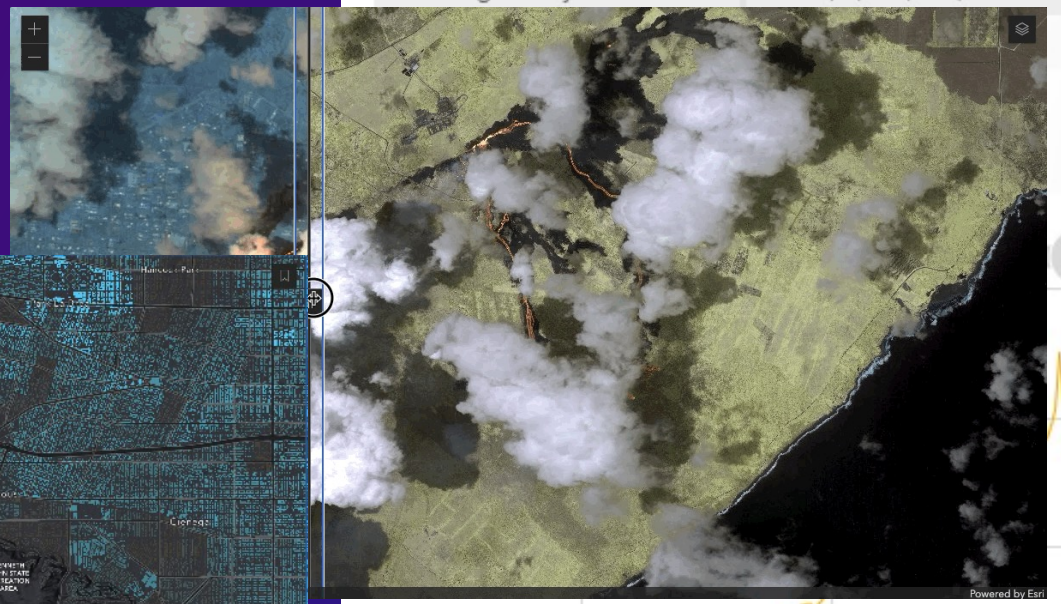
Source code

Interactivity

Anne Fitz

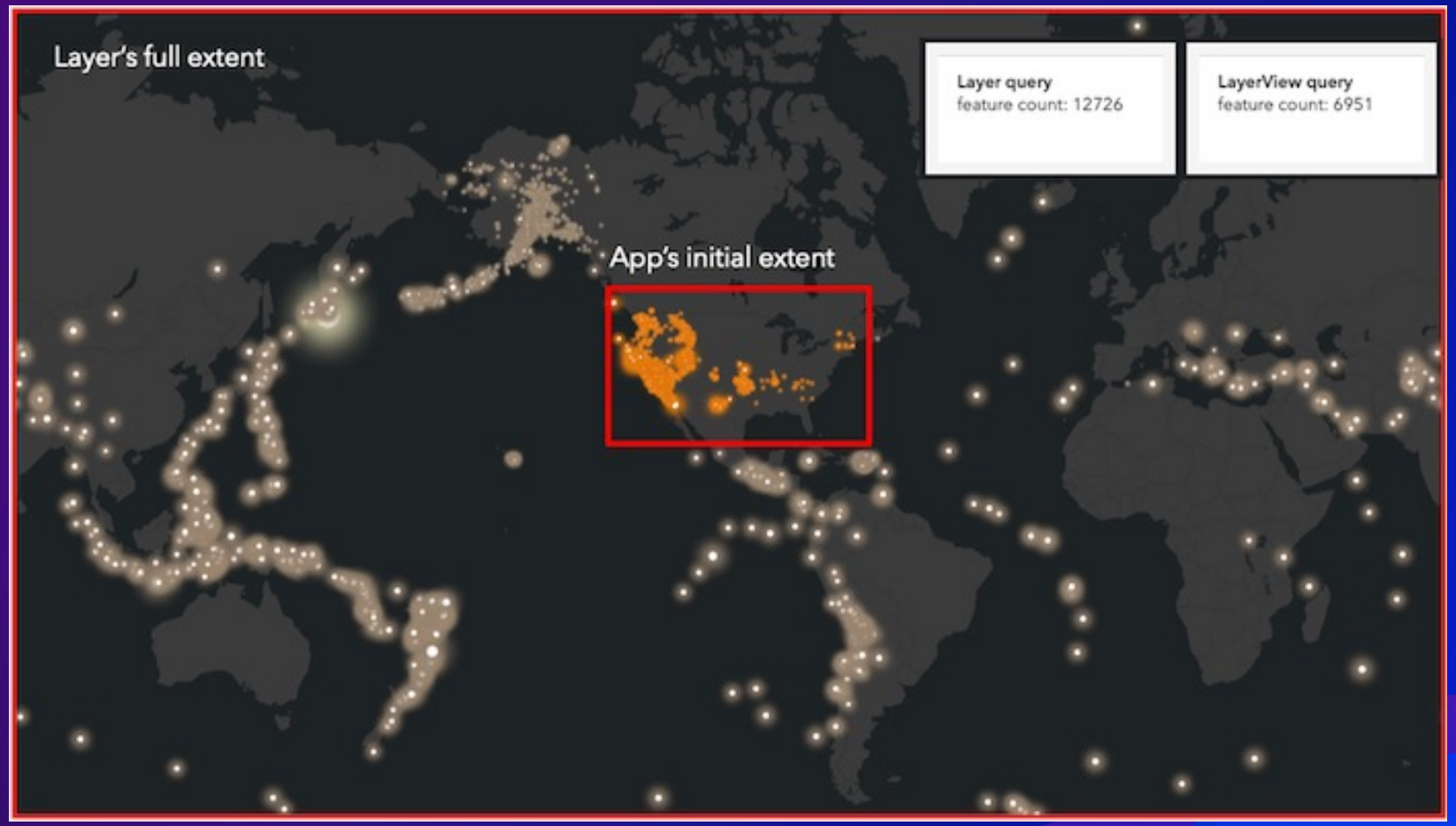
Interacting with your layer

- Popups
- Widgets
- Querying, highlighting, filtering



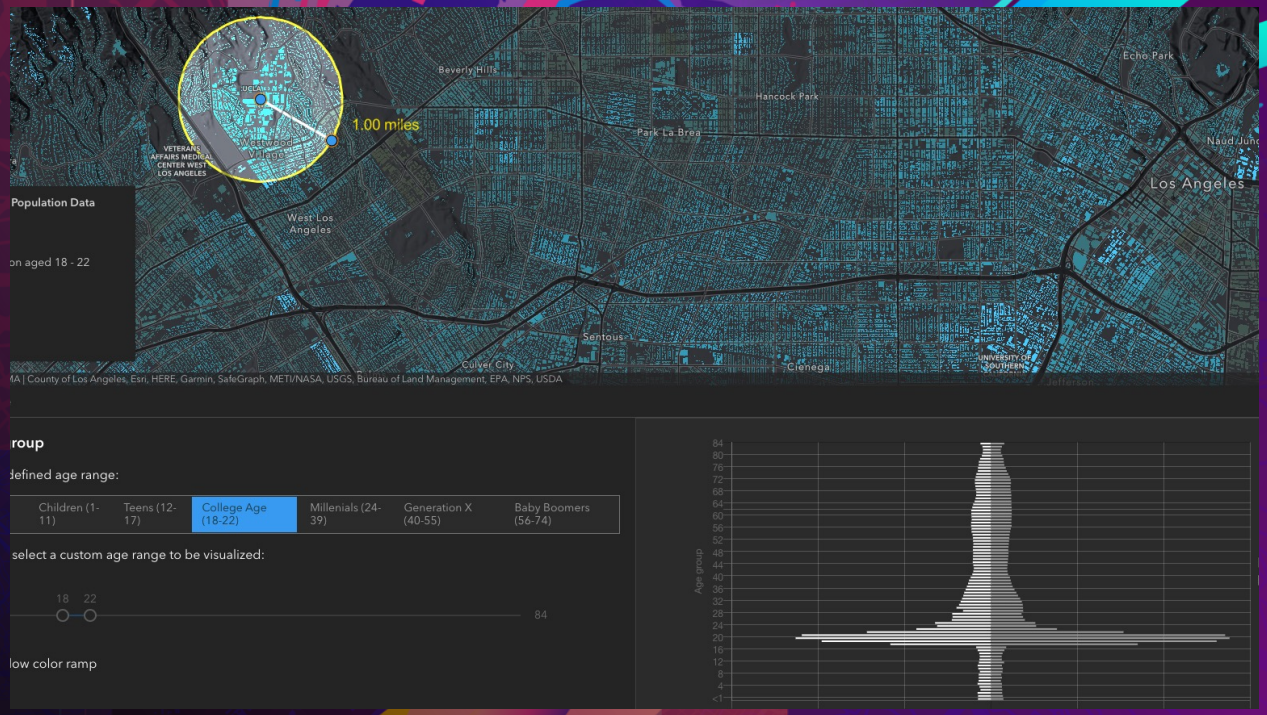
LayerView

- Created when a layer is added to a MapView or SceneView
- Renders features in the view
- Allows query, filtering, highlighting on the client-side



Querying

- Attribute queries
  - Only return features that pass a SQL WHERE clause
- Spatial queries
  - Only return features that pass a spatial filter
- Statistic queries
  - Return statistics about selected features (min, max, sum, etc…)



Age and Income in Los Angeles

Anne Fitz

Copyright © 2021 Esri. All rights reserved.