

# ArcGIS API for JavaScript: Working with Your Data Using the Feature Layer

Anne Fitz and Bjorn Svensson

slides: <https://arcg.is/1zevmz>

2021 ESRI  
DEVELOPER SUMMIT



# Introductions



Anne Fitz



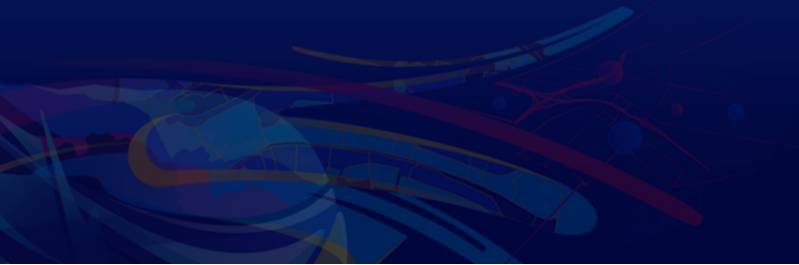
Bjorn Svensson

# Agenda

- Intro to FeatureLayer
- Performance
- Visualization
- Labeling
- Interacting with the FeatureLayer
  - Querying
  - Filtering
  - Highlighting
- Editing features

# What is the FeatureLayer?

- More than 20+ different layers
- FeatureLayer:
  - for query, visualize, analyze
  - clientside rendering
  - Service-based and file-based





# What's so special about FeatureLayer?

## FeatureLayer

[Constructors](#) | [Properties](#) | [Methods](#) | [Type definitions](#) | [Events](#)

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

Class: [esri/layers/FeatureLayer](#)

Inheritance: [FeatureLayer](#) → [Layer](#) → [Accessor](#)

Since: ArcGIS API for JavaScript 4.0

- Works with your data as features
- Can be used for editing
- Allows for dynamic styling and interactive workflows
- Support for client-side filtering, querying, and statistics

# Adding a FeatureLayer to your map

## Sources

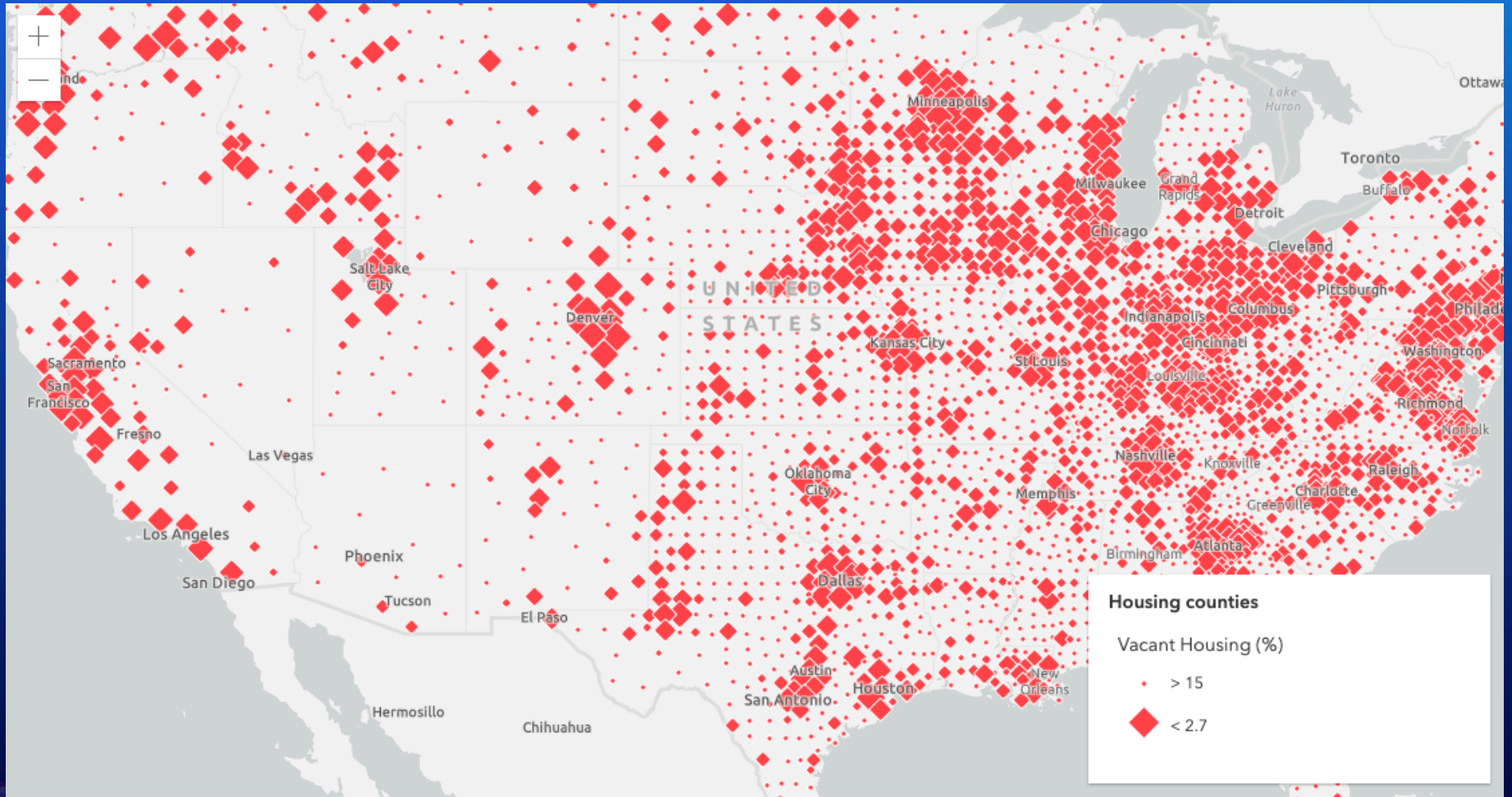
- Feature services or map services
- Feature collections
- Portal item (from ArcGIS Online or Enterprise)

```
const layer = new FeatureLayer({
  url: "https://<url to my server>/FeatureServer/0",
  // portalItem: {
  //   id: "item id from portal"
  // },
  //renderer: { ... },
  //popupTemplate: { ... },
});

map.add(layer);
```

# Adding a FeatureLayer to your map

*Vacant housing in the United States by county*



# FeatureLayer network performance





# API optimized for performance

- Compression: brotli
- Data format: prototype buffers
- Cacheability
  - consistent, predictable application queries
  - tiling
    - dynamic feature tiles
  - cache-control for caching in browser and CDN

demo



# Your settings for optimal cacheability

- disable editing on the feature layer
  - or create non-editable view layer
- set share level to everyone
- increase cache time

# Visualization: Rendering

A renderer defines how the FeatureLayer is drawn.

SimpleRenderer   ClassBreaksRenderer   UniqueValueRenderer

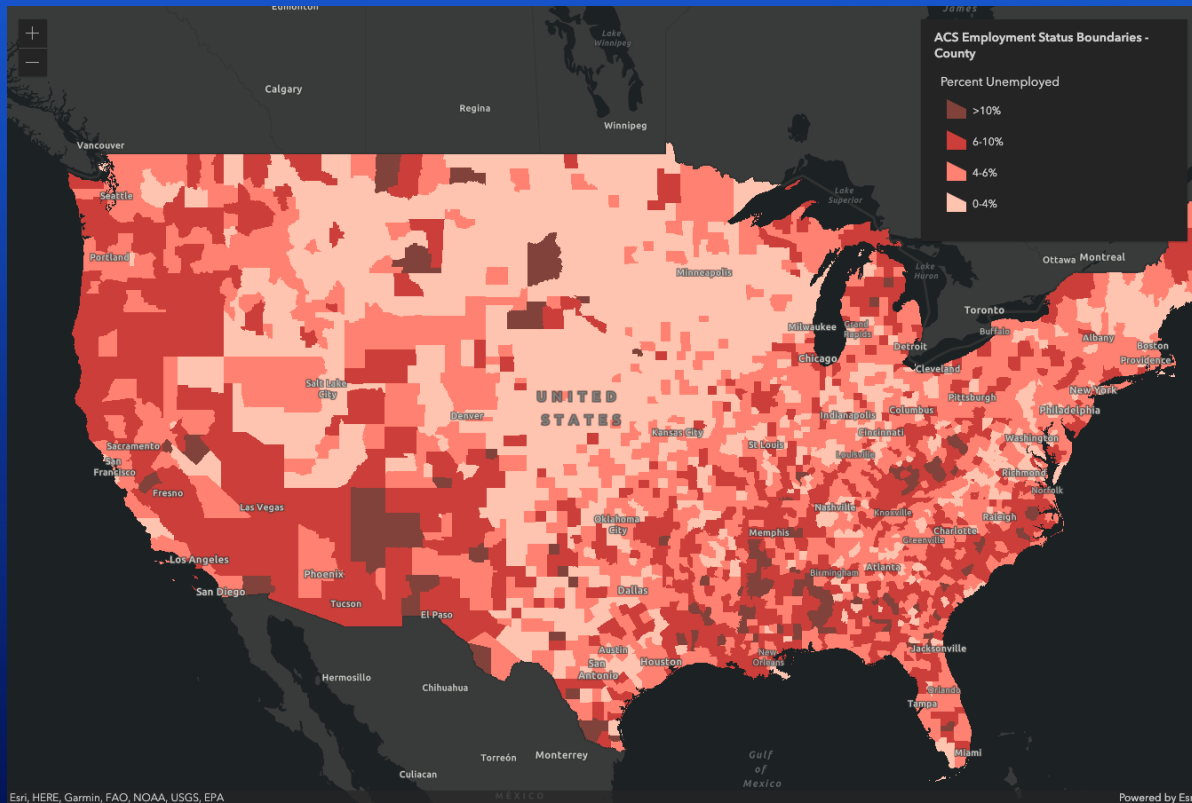
---

HeatmapRenderer   DotDensityRenderer   DictionaryRenderer

---



# Visualization: Rendering Demos

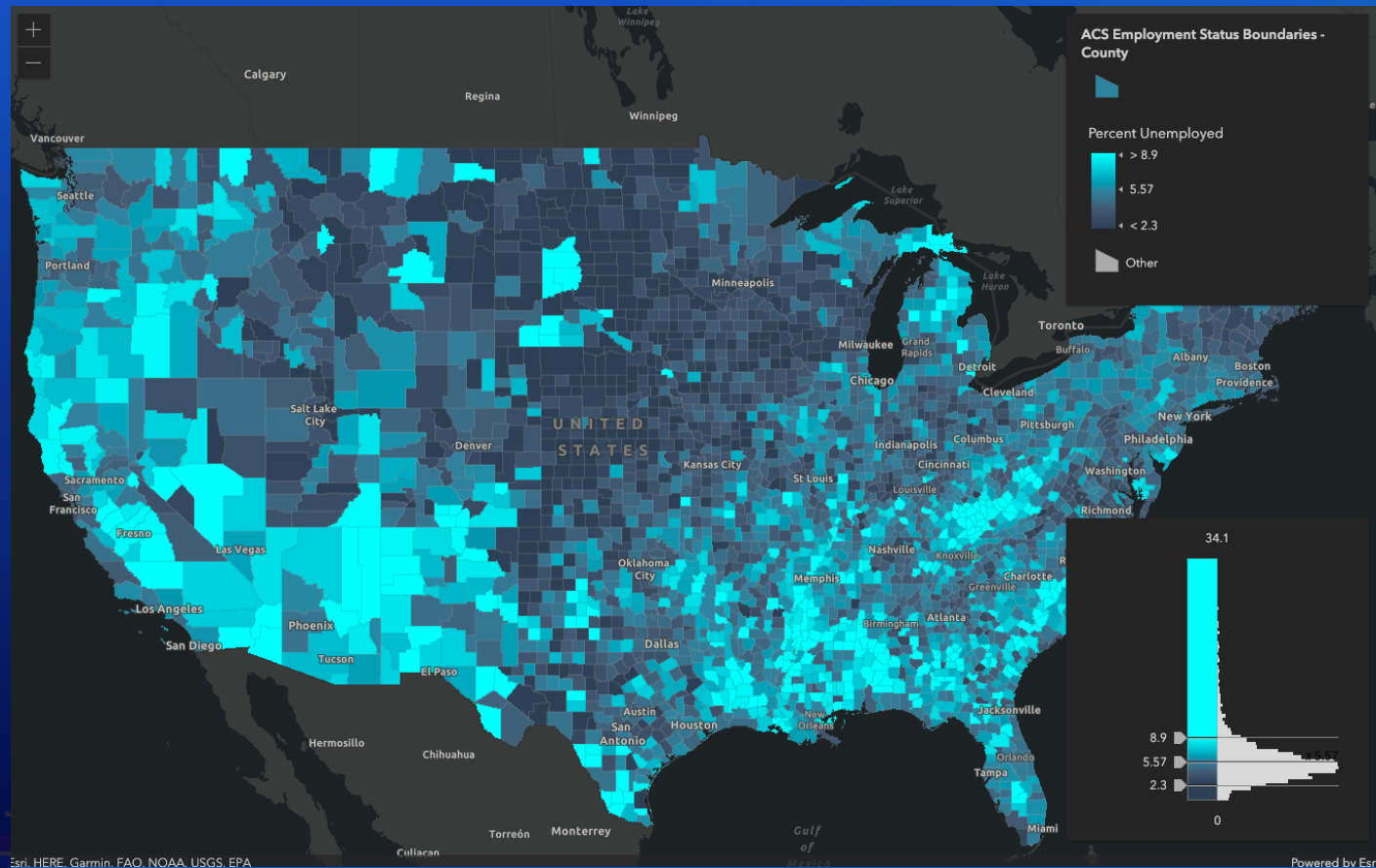


- Simple Renderer
- Class Breaks
- Visual Variables



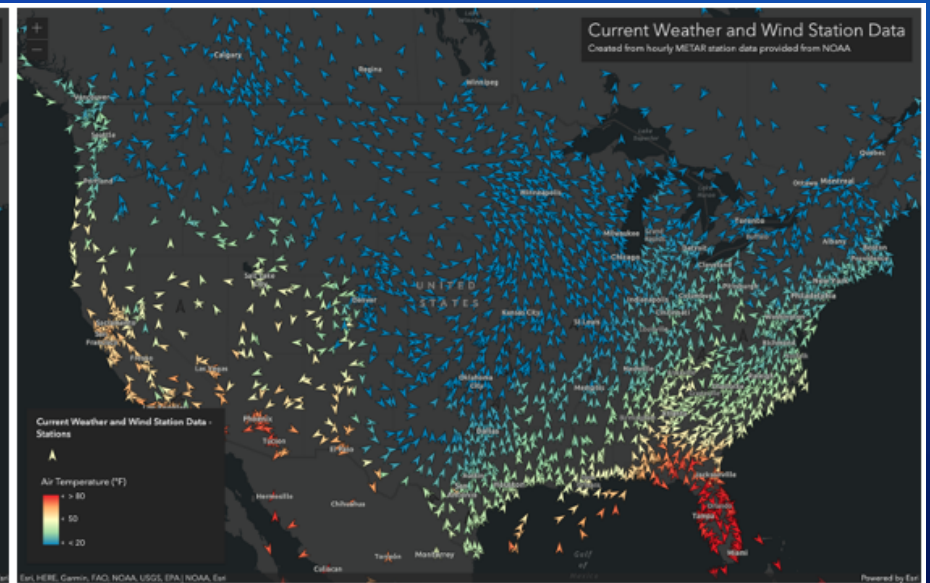
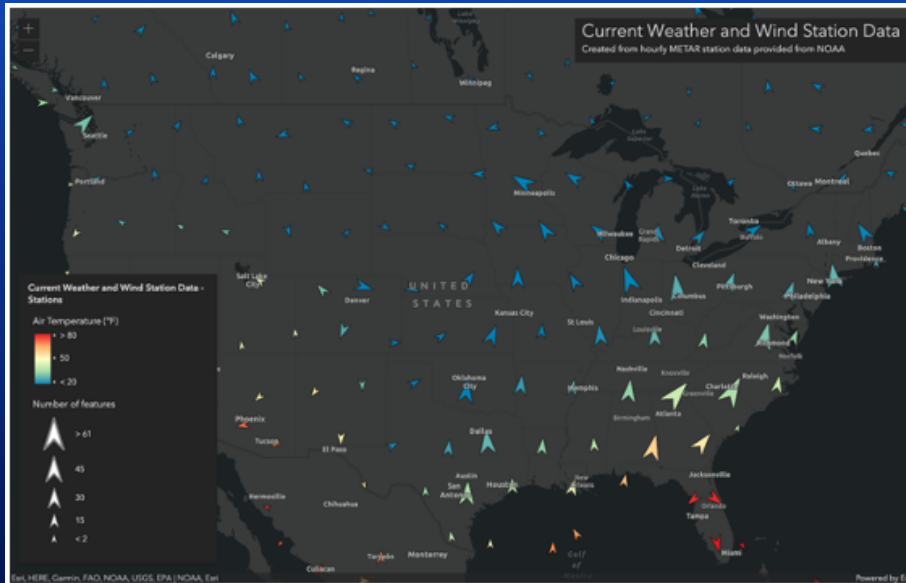
# Visualization: Smart Mapping

Smart Mapping APIs: generate renderers with "smart" default symbols based on the summary statistics of the dataset and the basemap



# Visualization: Clustering

**Clustering:** a method of reducing points by grouping them into clusters based on their spatial proximity to one another.



# Labeling

- FeatureLayer has labelingInfo and labelsVisible properties
  - labelingInfo is an array of LabelClass with
    - labelExpressionInfo,
    - labelPlacement,
    - maxScale and minScale,
    - where,
    - and symbol as TextSymbol or LabelSymbol3D
- or a PortalItem with labels defined

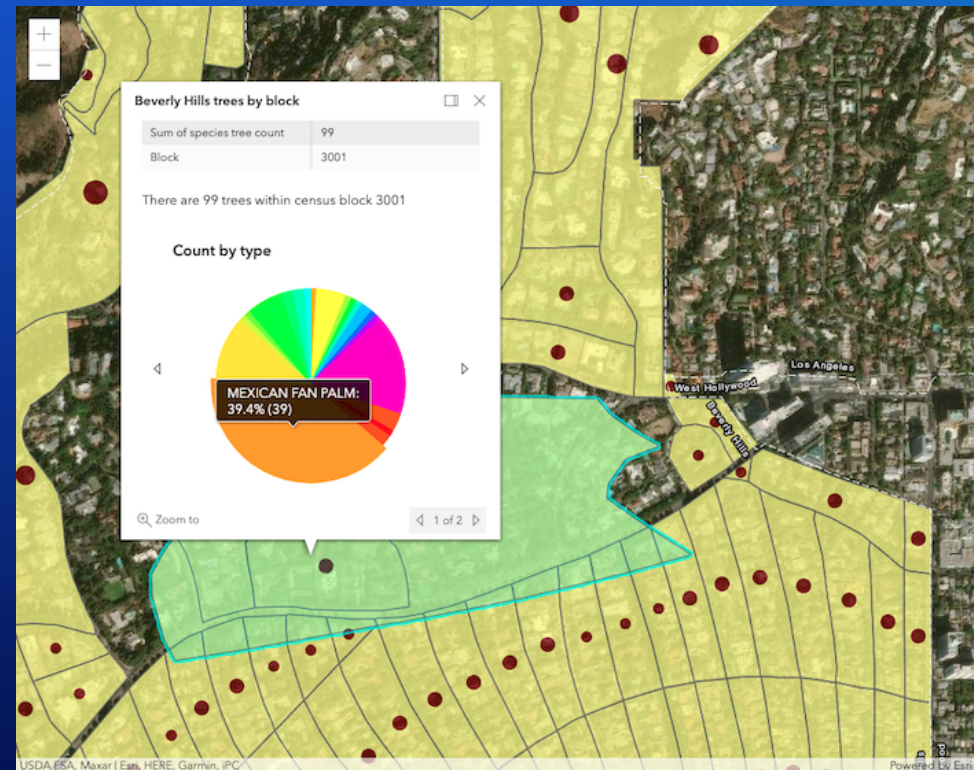
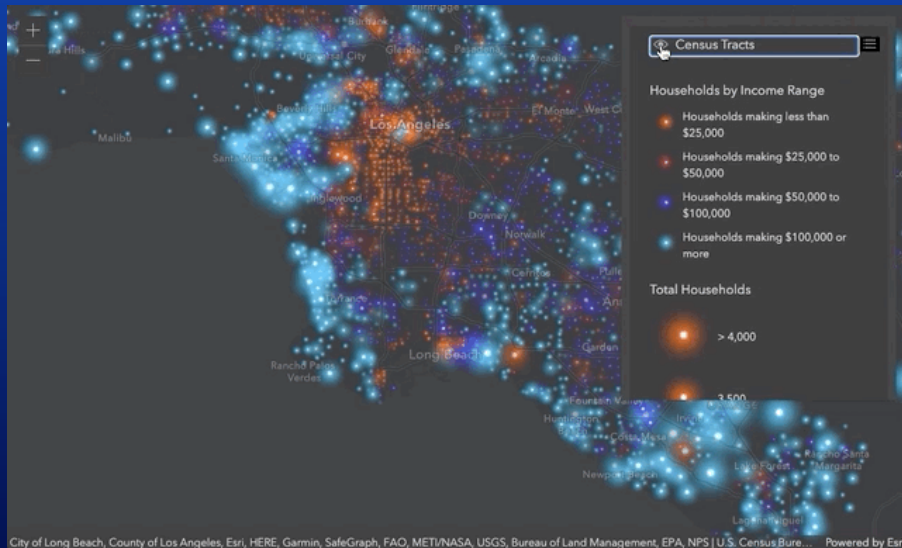
demo





# Interacting with your layer

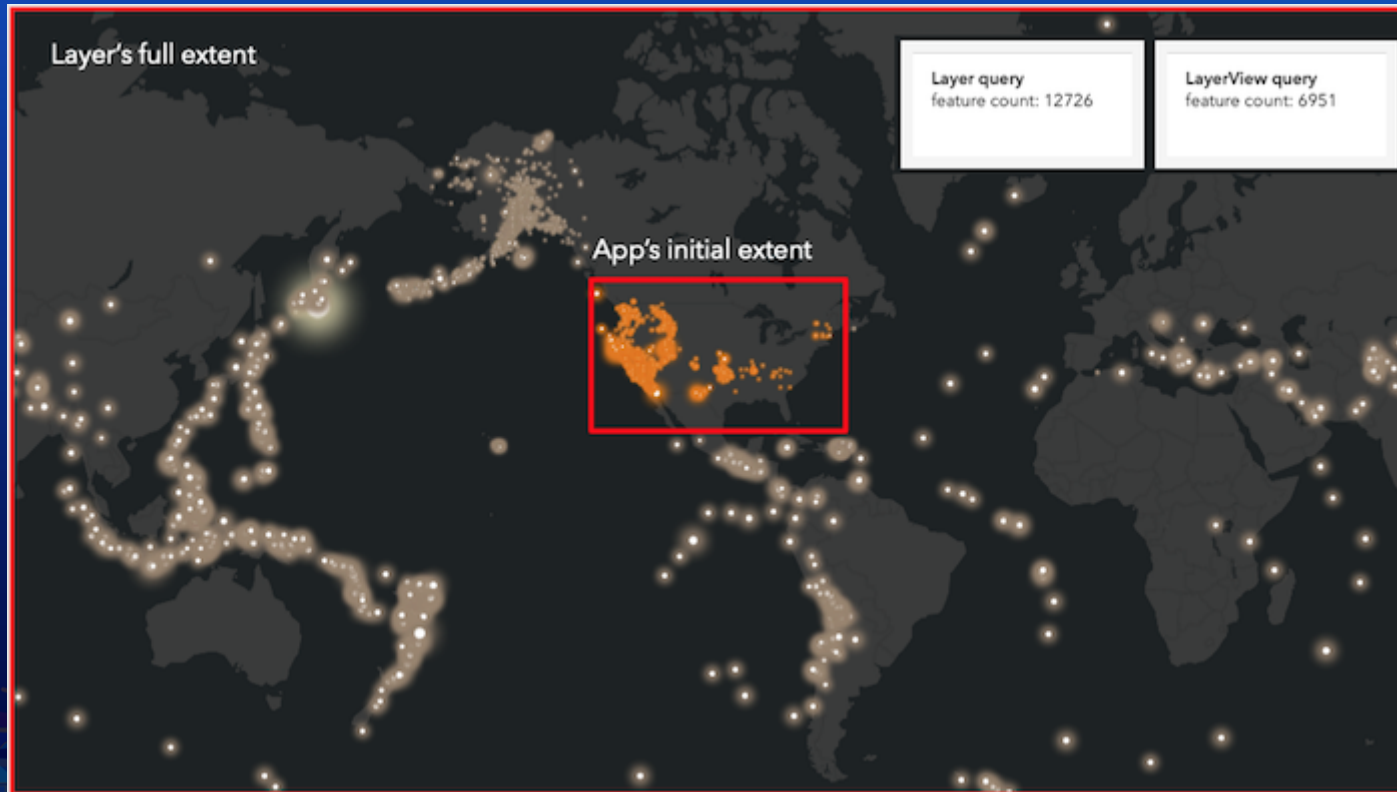
- Popups
- Widgets (Legend, LayerList, etc)
- Highlighting, querying, filtering





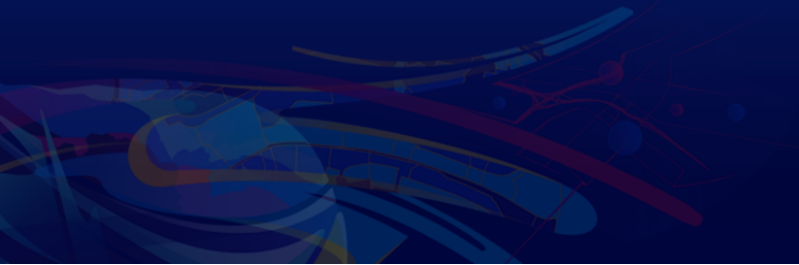
# Layer vs. LayerView

- LayerView
  - Created when a layer is added to a MapView or SceneView
  - Responsible for rendering features in the view
  - Allows query, filter, and highlighting on the client-side



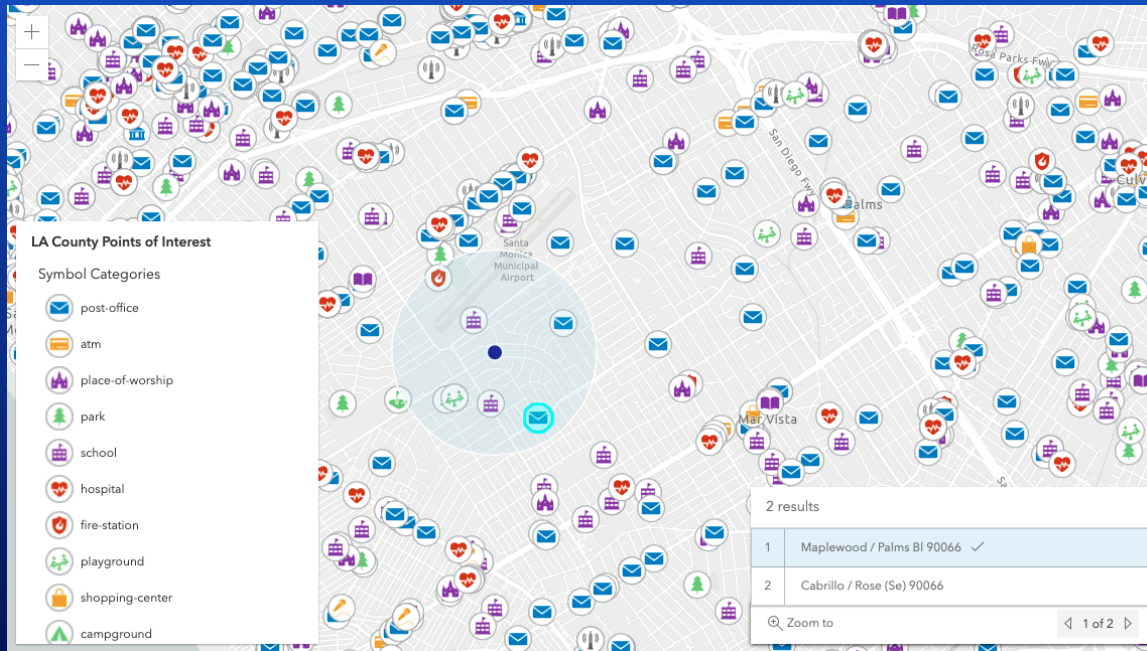
# Querying

- Attribute queries
  - select only features passing a WHERE SQL clause
- Spatial queries
  - select only features passing a spatial filter
- Statistic queries
  - returns statistics about the selected features



# Server-side querying

Bring features from your data into the web browser.



## Query features

```
featureLayer.queryFeatures({  
  geometry: point  
}).then(function(featureSet){  
  // do something with the results  
});
```

## Query attachments

```
featureLayer.queryAttachments()
```

## Query related features

```
featureLayer.queryRelatedFeatures()
```

# Client-side querying

Query data already in the web browser.

```
featureLayerView.queryFeatures ( )
```

- Really fast
- Avoids round-trips to the server
- Only works with available features
- Make sure you have all the attributes you need

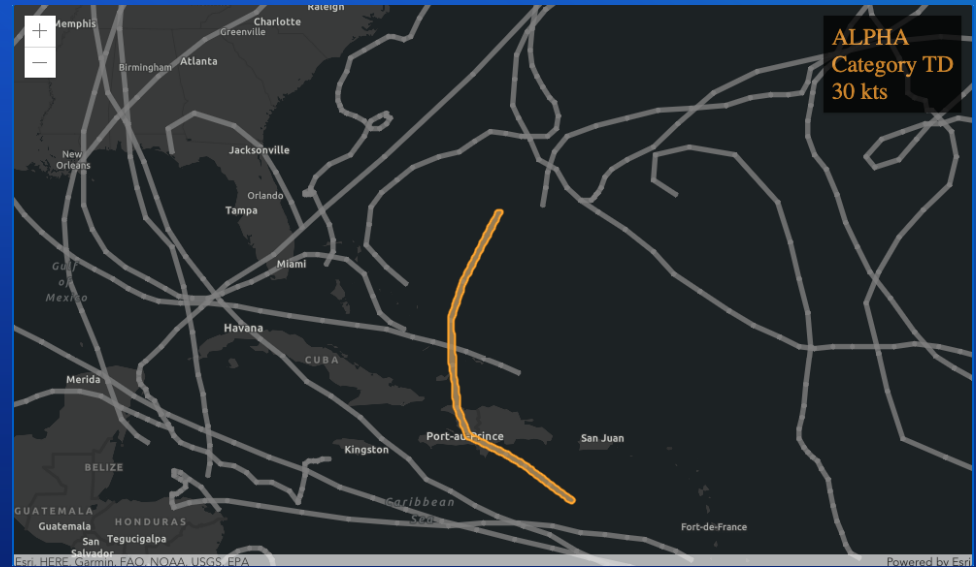
Demo

Demo - hover



# Highlight

- Highlight features on the LayerView
- Maintain a handle to the current highlight
- Highlight options: color, opacity, halo

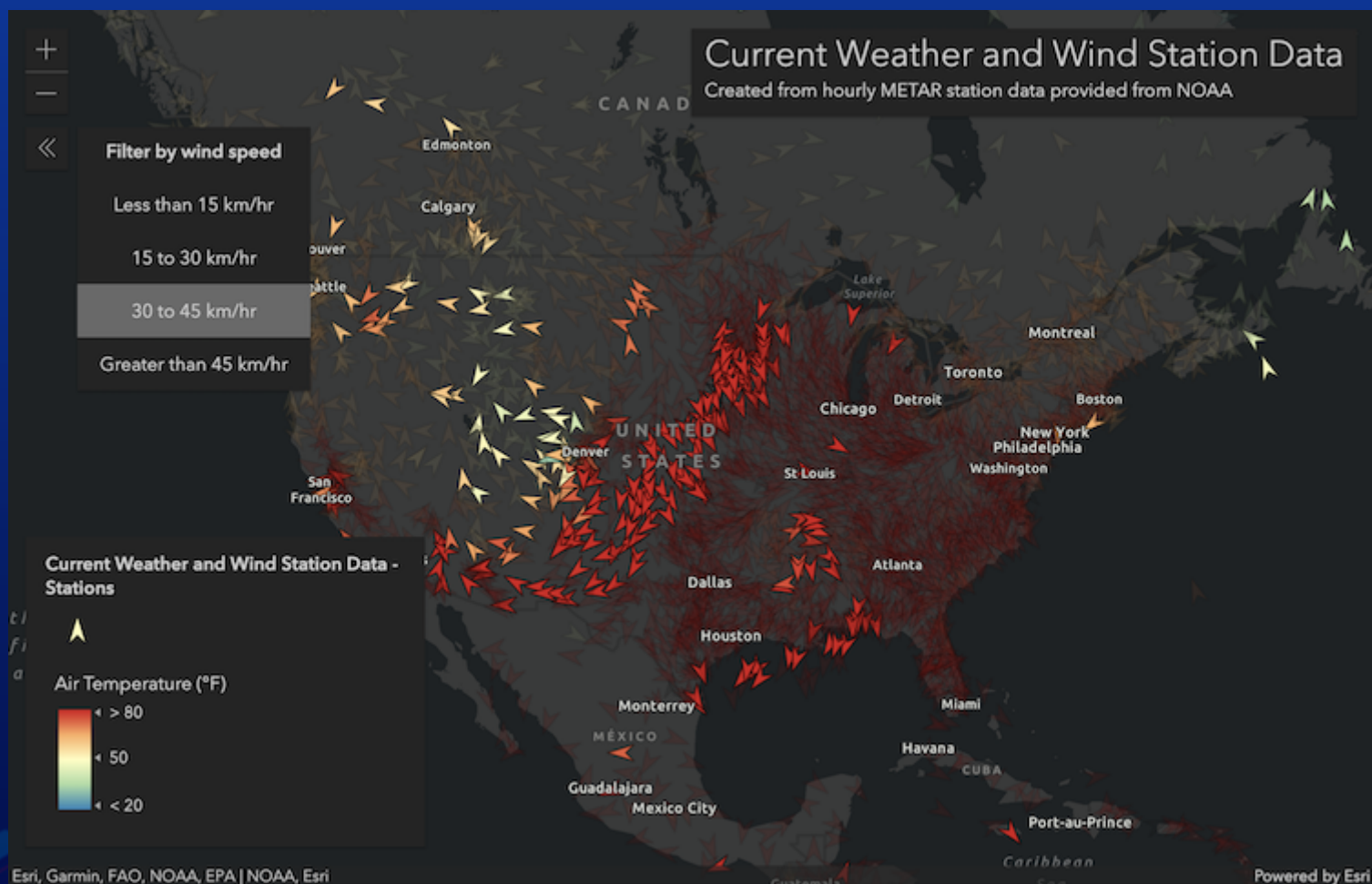


```
if (highlight){  
  highlight.remove();  
}  
highlight = layerView.highlight(result.features);
```



# Filtering

- Define the filter criteria
- Define the style for filtered features
- Apply the filter to the LayerView
- For server-side filtering, use `definitionExpression`



# Editing

An editable FeatureLayer

Feature Layer (hosted)

Editing

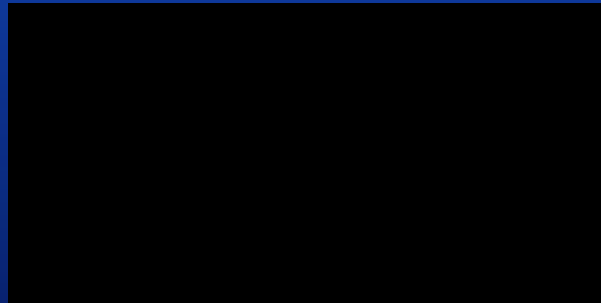
☒ Enable editing.

# Editing options


- Editor widget
- FeatureTable widget with editingEnable/editing properties
- FeatureLayer.applyEdits()
- FeatureForm widget & FeatureLayer.applyEdits()

# Editing demos

Samples







Please provide your feedback for this session  
by clicking on the session survey link directly  
below the video.



esri

THE  
SCIENCE  
OF  
WHERE