ArcGIS Runtime: Styling Maps

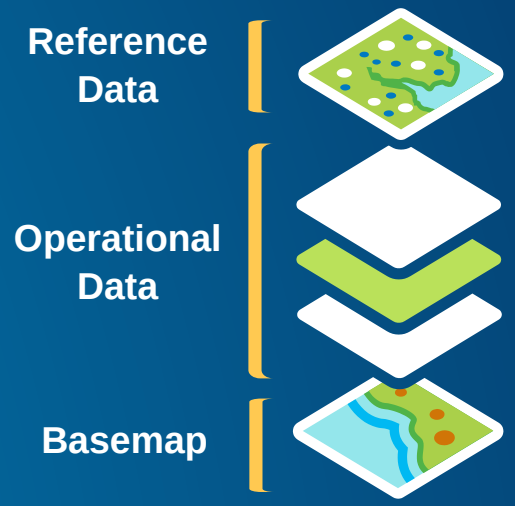
Ralf Gottschalk, Daniel Lee, Lucas Danzinger

Map Styling – What is this session about?

- Creating beautiful functional maps for your Runtime Apps
  - Not about cartography
- It is about how the decisions you make with
  - Layers
  - Symbols
  - Renderers
  - Labels
- Impact and are impacted by your
  - Authoring environment
  - Deployment environment

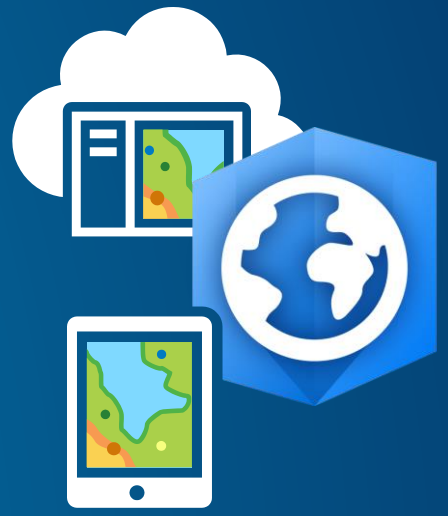
Styling Maps – Authoring Maps

- Basemaps
  - Size? Performance? Where does it come from? What is it's purpose?
- Reference Data
  - Do I need labels? Do they need to be dynamic?
- Symbology
  - Are there specific symbology requirements for my app?
    - Advanced verses Simple Symbology
    - For your basemap, operational layers, or both?
- Renderers
  - How does my data need to be displayed?
- Do I need tools to help me create a map?
- Do I already have a map and data I want to use?



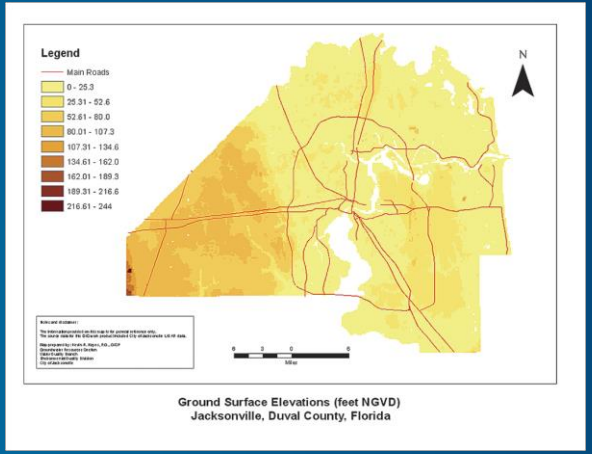
Styling Maps - Authoring Maps

- ArcGIS Online
  - Use Smart Mapping tools to create Webmaps
    - Easily create and edit beautiful and effective maps
  - Share and use these maps everywhere
  - Access to lots of existing maps, basemaps, and data to use for your app
- Runtime
  - Programmatically edit or create new Webmaps
  - Provides APIs to take maps offline
  - Use a map you have authored in Pro
- Pro
  - Use powerful cartographic and analysis tools to create and style data
  - Share as a Webmap everywhere
  - Share as Services many can be taken offline
  - Share as a Mobile Map Package with the Runtime



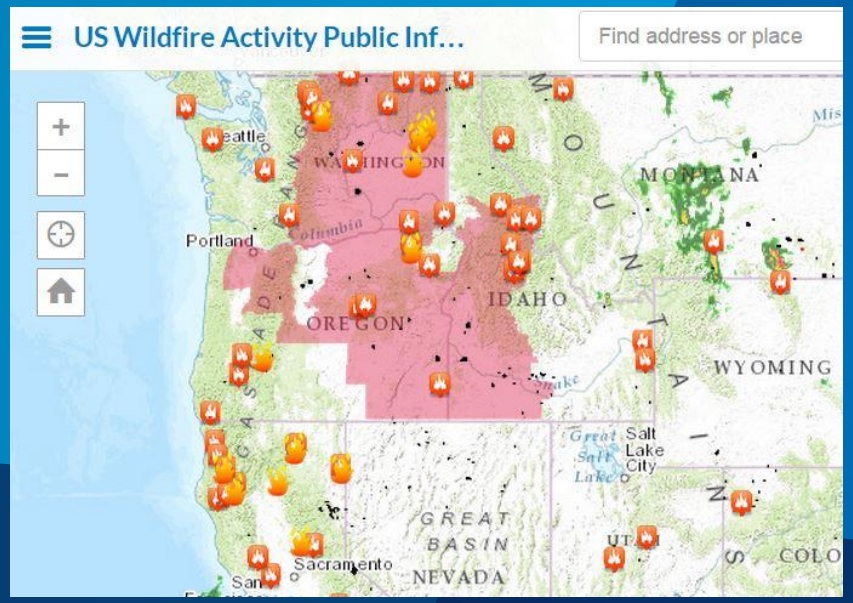
Styling Maps – More decisions

- What is my map going to do?
- Is this a:
  - Map for every app?
  - A map requiring complex symbols?
  - Will the app be connected to the internet?
    - Always, sometimes, or never?
  - A map that uses custom data?



Styling Maps – Authoring Maps

- 3 levels of functionality
  - Desktop / ArcGIS Pro
  - Runtime
  - Web
- When authoring a map ask yourself
  - Where will your map live?
  - How will the map be used?
  - Will your app use a map that
    - Can be accessed through all 3 tools?
    - Will just be used with Runtime?
- Thinking about the above will help determine what tool to use



Map for Every App

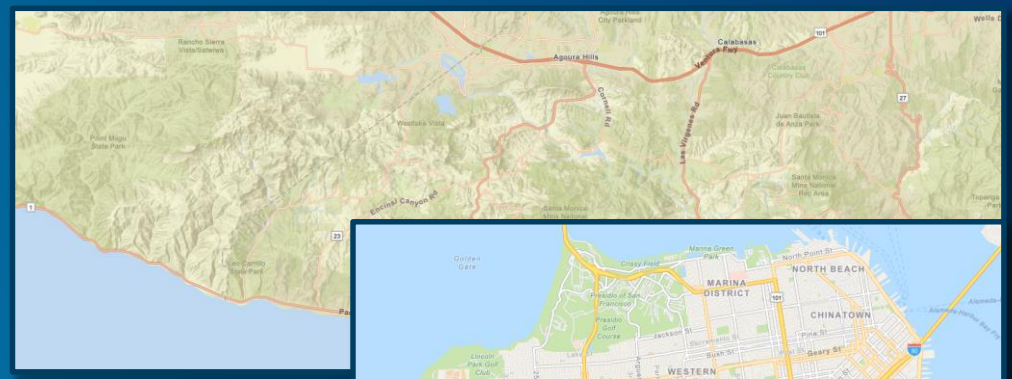
Daniel Lee

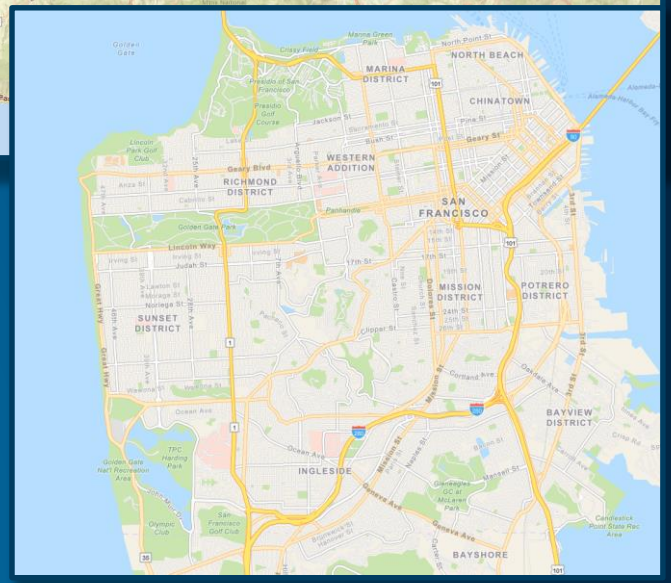
Summary – A Map for Every App

- Created a map that everyone has access to
  - Used ArcGIS Online
- Used the Smart Mapping tools to create symbols and renderers very easily
  - Fairly straight forward symbology requirements
- Chose to use a Vector Tile Layers for the Basemap
  - Simple map and wanted it to look great on all platforms

Basemaps

- Provide context for your app
  - Generally not primary focus of that map
- 3 to choose from in the Runtime
  - Raster Tile Layers
  - Vector Tile Layers
  - Mobile Map Package
- Which one do you choose?
  - Each basemap type is best suited for different use cases
  - Pick the one that is optimal for your map and your app



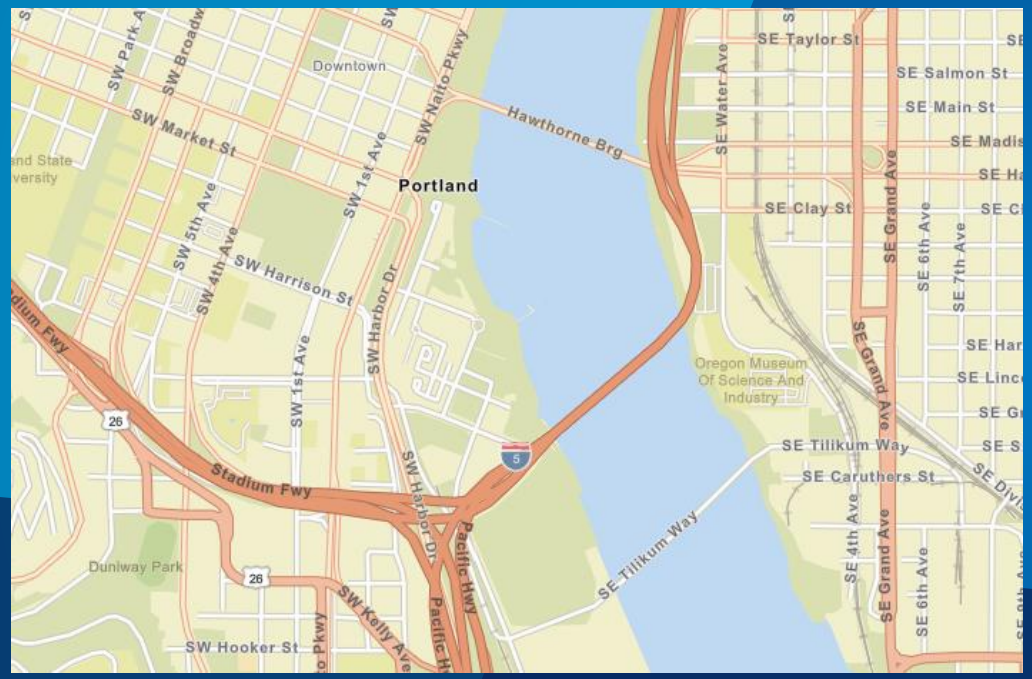


Basemaps – Raster Tile Layers

- Data is stored as images based on a tiling scheme
  - Cooked from an existing map or use the ones from ArcGIS Online
    - Online services or use a TPK
- Advantages
  - Rendering performance
    - No client side processing to required
  - Easy to author
    - Number of layers, complexity of the symbols, labels, data types all do not matter
  - Use ExportTileCacheTask to take the data offline
- Disadvantages
  - Large file size with long cooking times
  - Cooked at a specific DPI – quality could be compromised on high res devices
  - Everything is baked into the map
    - Points and labels will always rotate with the map

Basemaps - Vector Tile

- Data is stored as vectors based on tiling scheme
  - Cooked from existing maps or use the ones from ArcGIS Online
- Advantages
  - Vectors look great on high DPI devices
  - Small, fast cooking times
    - Indexed Vector Tile Layers are even smaller
  - Labels and points stay screen aligned
    - Labels use conflict detection and will declutter on the fly
- Disadvantages
  - Map authoring is critical for performance
    - Tiles must be processed on the client at render time
    - Number of layers and complexity of the data will impact on performance
    - There are limitations to data types, symbology, and renderers that you can use

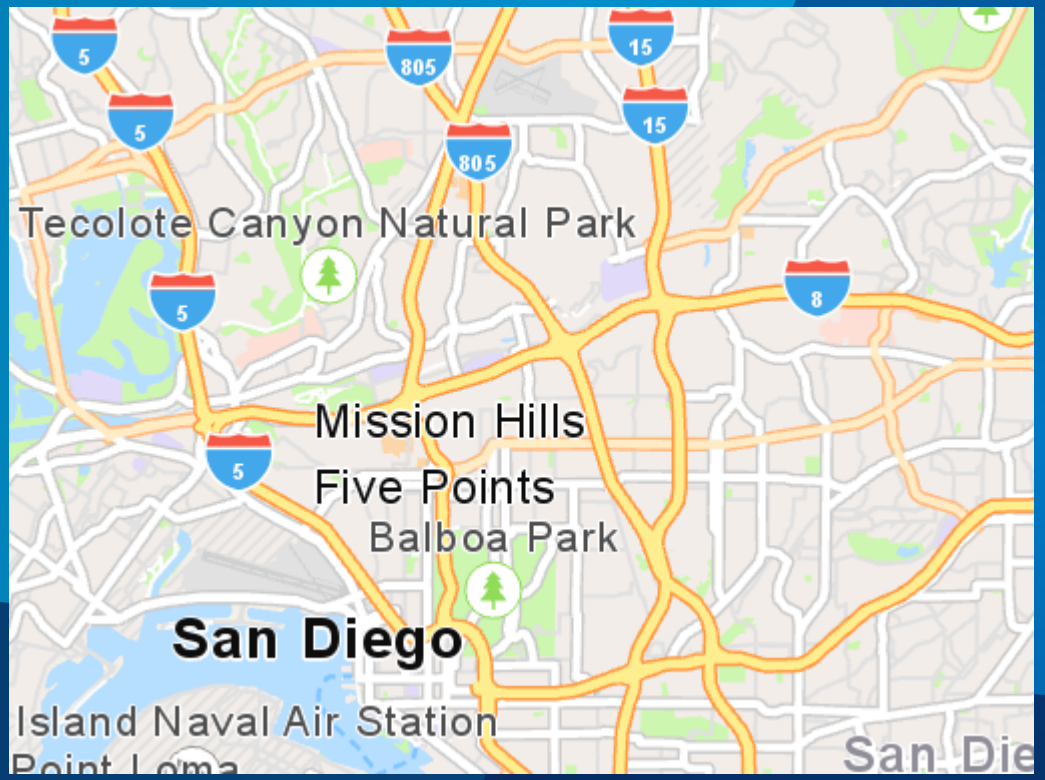


Raster vs Vector Tiles Demo

Lucas Danzinger

Basemaps – Mobile Map Package

- Packaged up ArcGIS Pro Map
  - Contains a basemap layer and operational layers
    - Packages can be purchased from Esri
- Advantages
  - Contains road networks and locators for searching and routing
    - Can be leveraged in completely disconnected workflows
  - Preserves my map
    - Symbology and label are made runtime compatible
- Disadvantages
  - Size can get large
  - Map authoring impacts performance (not as significant as vector tiles)
  - Only for Runtime and Pro



Advanced Symbols with Offline Capabilities

Lucas Danzinger

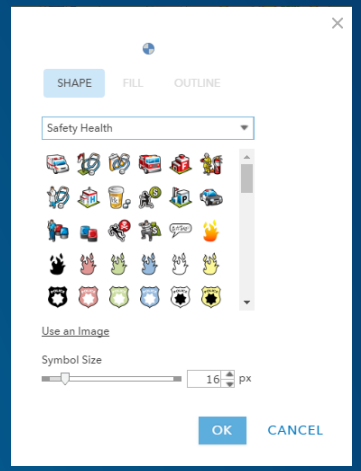
Summary – Advanced Symbols with Offline Capabilities

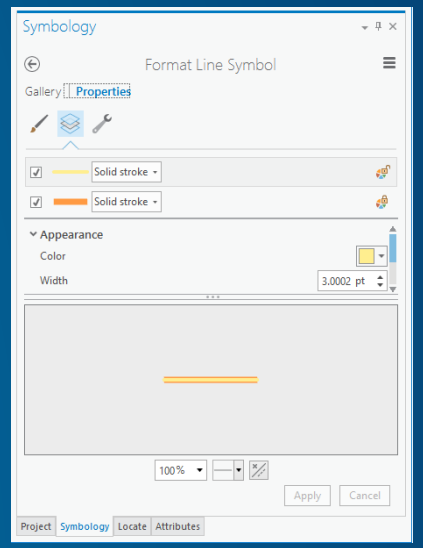
- Created a highly specialized map for a specific workflow
  - This map is only for people in his organization
- Required Advanced Symbology to represent the data properly
  - This way his clients know what they are working with
- Offline Requirements
  - Published Feature Services to generate an offline Geodatabase
  - Used his own imagery
    - Create local Tile Packages for offline use

Symbology

- Simple Symbols
  - Marker, Line, Polygon, and Picture Marker Symbols

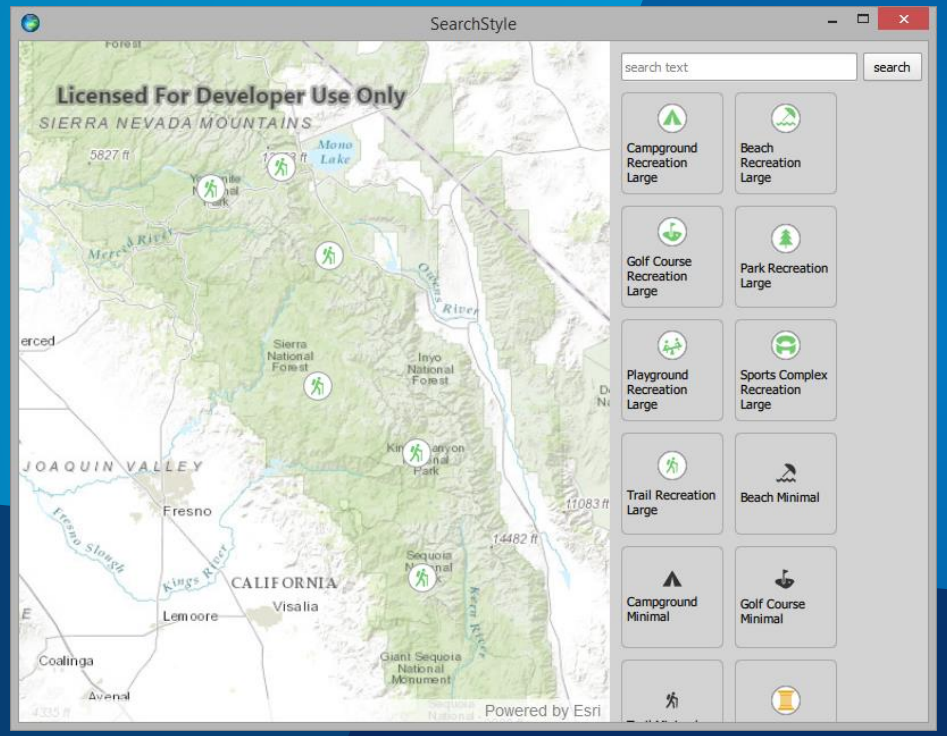
- Advanced Symbols
  - Desktop Symbology
  - Multilayer Vector Symbols
    - Pro Symbology in the Runtime





Symbology – Where to Author

- Simple Symbology
  - Online
    - Create with ArcGIS Online authoring tools
  - Runtime
    - Create Programmatically through Runtime APIs
  - ArcGIS Pro
    - Publishes both Simple and Advanced symbology for Webmaps and Feature Services
- Advanced Symbology
  - Authored in Pro
    - Enabled by default through the Runtime with Feature Services
    - Mobile Map Packages
    - Very soon these can be authored and shared through Style Files



Style File Demo

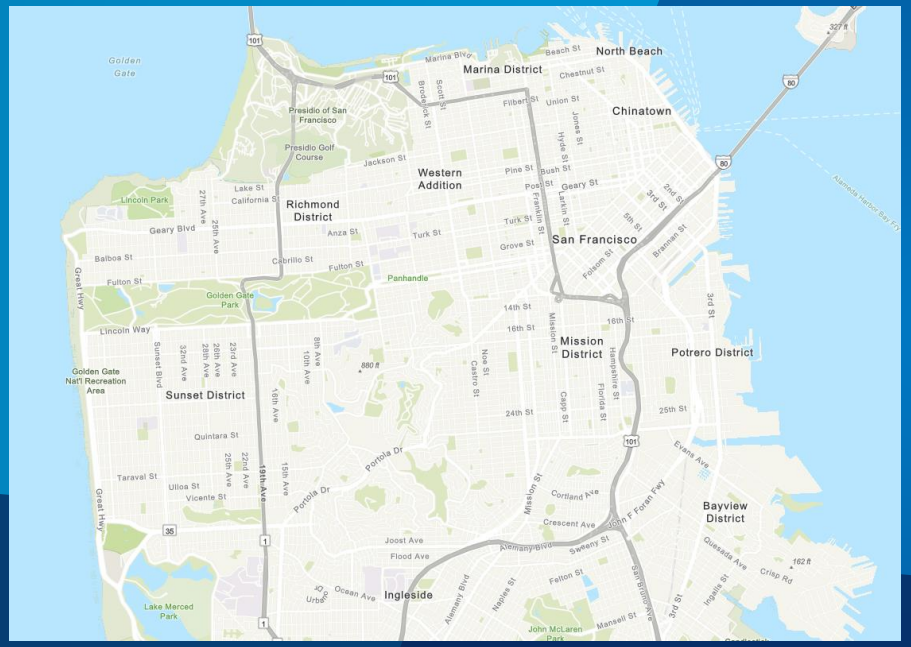
Ralf Gottschalk

Renderers – Authoring

- ArcGIS Online
  - Use Smart Mapping tools to easy author renderers without even knowing that you did it
    - Makes it easy to show patterns and make beautiful maps that can be used in any app
- Pro
  - Use powerful tools to create advanced visualizations
    - Analyzers that will tell you when something is not supported for your target environment
- Runtime
  - Provides read-only support for most of the renderers authored through the Webmap
  - Offers an API to allow developers to create and modify renderers
  - Provides programmatic support for many different feature and raster renderers

Renderers – When Creating Renderers for Runtime

- Visual Variables
  - Data driven visualizations
    - Color, Size, Opacity, and Rotation
  - The API behind Smart Mapping
- Create renders from Pro these are based on the Rest Spec
  - The additional Pro renderers that not in the rest spec further enhanced through Visual Variables
    - Example – Graduated Symbols Renderer = Web's Counts and Amounts
- Runtime currently has read-only support for Visual Variables
  - Must be authored through the Web or Pro
  - API for Visual Variables will be coming soon



Specialized Map for an Offline App

Lucas Danzinger

Summary – Specialized Map for an Offline App

- Map must work entirely offline
- Utilizes Routing
- Map needs to look like it does in Pro
- The Mobile Map Package provides the basemap and operational layers

ArcGIS Map Image Layers

- Don’t forget about ArcGIS Map Image Layers
- You can leverage additional not support by your target platform
- These are maps dynamically created by the server
- Use Map Image Layers
  - To show Advanced Symbols
  - Uses Maplex Labeling

Authoring Maps through the Runtime – The Programmatic Map

- Runtime provides APIs to create and edit maps
- Use this workflow to
  - Implement custom data sources
    - Do this through Feature Collection Tables or Graphics Overlays
  - Add raster data to your maps
- You can mix and match all previous workflows and modify any type of map programmatically

```java
// a feature collection table that creates point geometry
FeatureCollectionTable pointsTable = new FeatureCollectionTable(pointFields, GeometryType.POINT, WGS84);

// set a default symbol for features in the collection table
SimpleMarkerSymbol markerSymbol = new SimpleMarkerSymbol(Style.TRIANGLE, 0xFFFF0000, 18);
SimpleRenderer renderer = new SimpleRenderer(markerSymbol);
pointsTable.setRenderer(renderer);

// add feature collection table to feature collection
featureCollection.getTables().add(pointsTable);

// create feature using the collection table by passing an attribute and geometry
Map<String, Object> attributes = new HashMap<>();
attributes.put(pointFields.get(0).getName(), "Current Location");
Point point = new Point(-79.497238, 8.849289, WGS84);
Feature addedFeature = pointsTable.createFeature(attributes, point);

// add feature to collection table
pointsTable.addFeatureAsync(addedFeature);
}
```

Programmatic Map

Daniel Lee

Summary – Programmatic Map

- Used and existing published map a basemap
- Custom data requirement so he used Feature Collection table

Tips and Tricks

- 3 Levels of Symbol Complexity
- Web
  - Simple Symbology only
    - Complex markers become Picture Markers at 96 dpi
    - Complex multilayer lines and polygons become simple
- Runtime is a subset of Pro Symbology
  - Vectorized Markers
    - Character marker symbols are based off of fonts
      - The font might not exist on the platform
    - Marker Geometries are Densified (Not True Curves yet!)
    - RGB only
    - Some symbols like EMF are not supported

Tips and Tricks

- Text Symbols
  - Are not vectorized
- Care must be take as to which font is you choose when authoring map
    - Various platforms do not have the same fonts
    - Different platforms have different rules for installing fonts
- Runtime will leverage the native system to find a suitable alternative
  - But your map might look different
  - Finding replacement fonts is a performance hit

esri

THE SCIENCE OF WHERE