ArcGIS Runtime: Styling Maps
Lucas Danzinger and Michael Wilburn
What is this session about?

- Creating *functional* maps for your Runtime Apps

- About the decisions you make with:
  - Layers
  - Symbols
  - Renderers
  - Authoring tools
What type of app are you building? What information are you showing? Who is your target audience?

1. **Simple location-based app**
   - To be run on mobile phone
   - Contains simple symbols, renderers, etc

2. **Heavy GIS user app**
   - May need to contain complex symbols
   - Aimed at Utility, Public Safety, Defense, or Intelligence sectors
   - Can be taken offline

3. **System integrator enterprise app**
   - Apps utilize custom data feeds, workflows, etc
   - Already existing apps, processes, etc. - just need to add a map
   - Limited or no access to the platform
Avenues for Authoring Maps

**Online**
- Use Smart Mapping tools to easily create/edit beautiful and effective web maps
- Share and use these maps everywhere
- Access to lots of existing maps, basemaps, and data to use for your app

**Pro**
- Use powerful cartographic and analysis tools to create and style data
- Share as a web map everywhere
- Share as services (many can be taken offline)
- Share as a Mobile Map Package with the Runtime

**Runtime**
- Programmatically edit or create new web maps
- Provides APIs to take maps offline
- Use a map you have authored in Pro
Persona #1
Simple location-based app
Summary – Simple location-based app

• Created a map that everyone has access to
  - Hosted as a web map on the web
  - Accessible by many clients

• Used Smart Mapping tools to create symbols and renderers
  - Helps you to very easily make interesting and cartographically-pleasing maps

• Chose to use a Vector Tile Layers for the basemap
  - Looks nicer on mobile phone (scales to all screen resolutions)
  - Smooth animation, labels rotate
  - Used Esri-provided data

• Simple authoring and coding experience
Persona #2
Heavy GIS user
Summary - Heavy GIS user app with complex symbols

- **Created a map that workers have access to (offline)**
  - Saved as Mobile Map Package
- **Used the advanced Pro symbol styling tools**
  - Create and modify complex multi-layer symbols
- **Organizational data used for basemap**
  - Points, lines, and vector tile package
Symbology

• Simple Symbols
  - Includes marker, line, and fill symbols
  - Examples:
    - **Point**: circle, triangle, square, and picture marker
    - **Lines**: solid line, dashed line
    - **Polygons**: solid fill, picture fill

• Advanced Symbols
  - Desktop Symbology Model
  - Supported in MMPK, Feature Service, Style Files
  - Multi-layer vector symbols
  - Examples:
    - Custom symbol (e.g. blue circle with a black outline that has a triangle in middle and a text offset)
    - Line with directional arrow at each vertex
    - Fill with custom marker symbol
<table>
<thead>
<tr>
<th>Symbology</th>
<th>Online</th>
<th>Pro</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Symbology</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Advanced Symbology</td>
<td>✗️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
## Symbology - Where to Author

<table>
<thead>
<tr>
<th></th>
<th>Online</th>
<th>Pro</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Simple Symbology</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Advanced Symbology</strong></td>
<td>❌</td>
<td>✓</td>
<td>❌*</td>
</tr>
</tbody>
</table>

*new capabilities for authoring advanced symbology coming to Runtime soon!*
Persona #3
System integrator
custom enterprise app

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

// set a default symbol for features in the collection table
SimpleMarkerSymbol markerSymbol = new SimpleMarkerSymbol(Style.TRiangle, BufferColor.BLACK, 10);
SimpleRenderer renderer = new SimpleRenderer(markerSymbol);
pointsTable.setRenderer(renderer);

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

// create feature using the collection table by passing an attribute and geometry
Map<String, Object> attributes = new HashMap<>();
attributes.put(pointFieldIdx, getName());
Point point = new Point(-70.407238, 8.849289, WGS84);
Feature addsFeature = pointsTable.createFeature(attributes, point);

// add feature to collection table
pointsTable.addFeatureAsync(addsFeature);
```
Summary – System Integrator Enterprise App

• They have existing app, processes, data formats
  - Just need to add in location/spatial capabilities to their app
• Used an Esri provided basemap
• Built the map up in code
• Parse custom data source and display as Graphics
• Create Simple Symbols and Renderers in code
Styling Maps takeaways

• There are many options (tools, workflows, data types, etc)
• Consider the 3 personas we highlight
• Use existing tools (AGOL, Pro) if you can
  - Useful UI/UX to aid you in styling maps
  - Often more robust options for customization than what is available in the API
  - You can update data without recompiling/re-releasing your app
• You can also style maps programmatically
  - No dependency on other tools
  - Potentially more flexible for fitting into existing custom workflows
  - Less styling options available, but we are working on improving this