ArcGIS API for JavaScript
Client-side queries, filters and analysis (without a server)

Undral Batsukh – ubatsukh@esri.com

Richie Carmichael – rcarmichael@esri.com

2021 ESRI DEVELOPER SUMMIT

Agenda

- **Client-side Layers**
  - **FeatureLayer, CSVLayer and GeoJSONLayer**

- **Client-side Queries**
  - **Layer vs. LayerView**

- **Client-side Filtering**
  - **Filters and Effects**

- **Client-side Analysis**
  - **Geometry Engine, Projection Engine and GeodesicUtils**

Client-side layers

- CSVLayer
- GeoJSONLayer
- FeatureLayer with feature collections

Client-side layers

- Fetch all features at once and store them on the client

- Uniform API

- Responsive and fast performance
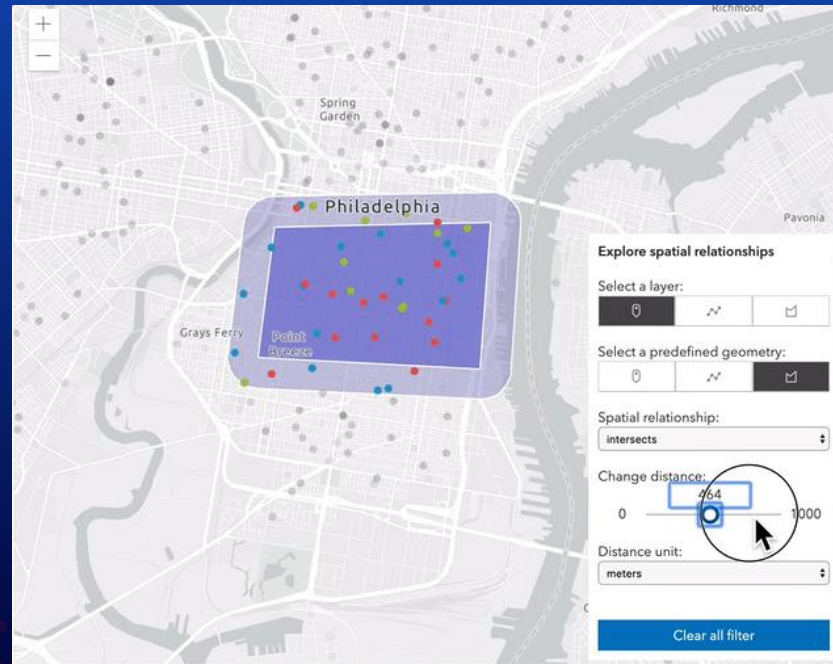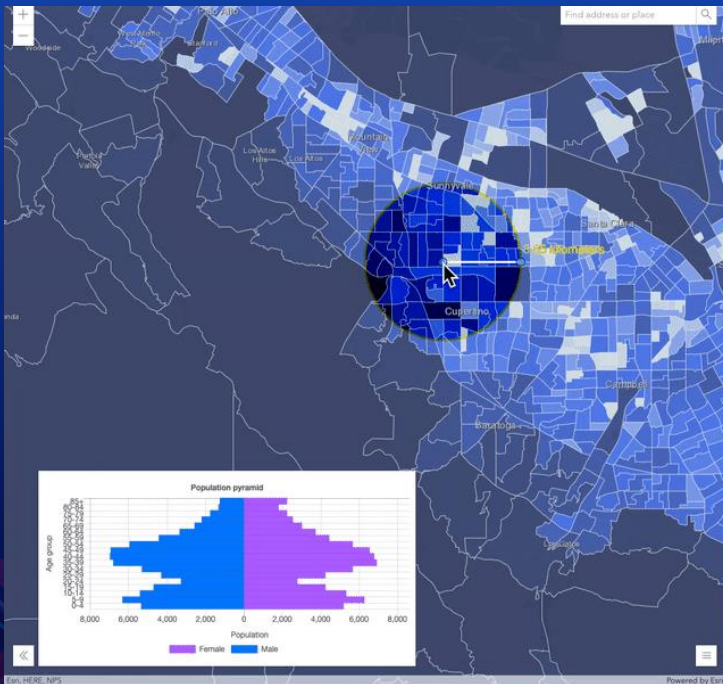
CSVLayer

- Add data from csv/txt file as *points*

- API Reference and SDK samples

```javascript
const csvLayer = new CSVLayer({
  url: "https://earthquake.usgs.gov/earthquakes/.../2.5_week.csv",
  copyright: "USGS Earthquakes",
  // SR in which the data will be stored
  spatialReference: { wkid: 102100 },
  delimiter: ",",
  latitudeField: "lat",
  longitudeField: "lon",
  // defaults to "__OBJECTID"
  objectIdField: "myOid",
  // create timeInfo for temporal visualization
  timeInfo: {
    startField: "time", // name of the date field
    // set time interval to one day
    interval: { value: 1, unit: "days" },
  }
});
```

CSVLayer tips

- X, Y coordinates must be in WGS84 in csv file.

- Specify the layer's spatial reference to match the view's spatial reference to improve the performance.

- Not supported:
    - No z-values support.
    - Cannot add, remove or update features.

CSVLayer tips

- Can pass the data by a blob url.

```javascript
const csv = `first_name|year|latitude|longitude
  Undral|2020|40.418|20.553
  Richie|2018|-118|35`;

const blob = new Blob([csv], {
  type: "plain/text"
});
let url = URL.createObjectURL(blob);

const layer = new CSVLayer({
  url: url
});
await layer.load();

URL.revokeObjectURL(url);
url = null;
```
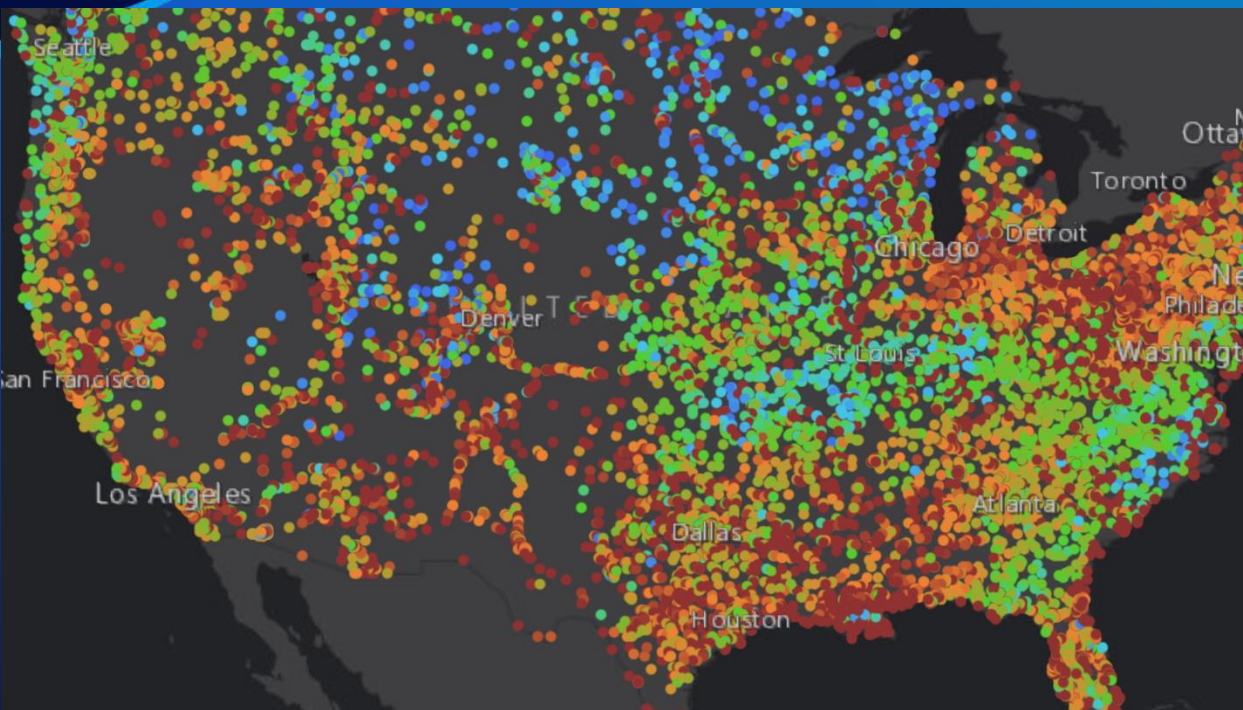
FeatureLayer

- Add client-side graphics by setting *FeatureLayer.source*
- API Reference and SDK samples

```javascript
const layer = new FeatureLayer({
  source: [
    new Graphic({ attributes: { myOid: 1 }, geometry: { ... } })
    new Graphic({ attributes: { myOid: 2 }, geometry: { ... } })
    new Graphic({ attributes: { myOid: 3 }, geometry: { ... } })
  ],
  // can be inferred from geometries
  geometryType: "point",
  // can be inferred from geometries
  spatialReference: { wkid: 2154 },
  // can be inferred from fields w/ field.type "oid"
  objectIdField: "myOid",
  fields: [
    new Field({ name: "myOid", type: "oid" })
  ]
});
```

Client-side FeatureLayer

FeatureLayer tips

- Supports point, polyline and polygon data in any spatial reference.

- Specify source only at the time of initialization.

- Use FeatureLayer.applyEdits() to add, remove or update features at runtime.

- Call FeatureLayer.queryFeatures() get the updated features at runtime.

GeoJSONLayer

- Add GeoJson data that comply with the RFC 7946 specification
- API Reference and SDK Samples

```js
const geoJSONLayer = new GeoJSONLayer({
  url: "https://earthquake.usgs.gov/earthquakes/.../all_month.geojson",
  copyright: "USGS Earthquakes",
  // SR in which the data will be stored
  spatialReference: { wkid: 102100 }
});
```

GeoJSONLayer tips

- Specify layer's spatial reference for performance.

- Support for *Feature* and *FeatureCollection*

- GeoJSONLayer.applyEdits to add, delete or update features.

- Not supported:
    - Mixed geometry types for consistency with other layers.
    - crs object - only geographic coordinates using WGS84 datum (long/lat)
    - No Antimeridian crossing
    - Feature id as string

GeoJSONLayer tips

- Create a blob url from GeoJSON object.

```javascript
const geojson = `{
  type: "FeatureCollection",
  features: [{
    type: "Feature",
    geometry: { type: "Point", coordinates: [-100, 40] },
    properties: { name: "none" }
  }]
}`;

const blob = new Blob([JSON.stringify(geojson)], {
  type: "application/json"
});
let url = URL.createObjectURL(blob);
const layer = new GeoJSONLayer({ url });

await layer.load();
URL.revokeObjectURL(url);
url = null;
```

Client-side layer tips

- Each implementation uses client-side query engine.
- Prefer GeoJSON over CSV.
- Set the layer spatial reference for performance.
- Proper attribution using *copyright* property.

Query

Query

- Query expressions are used to select a subset of features and table records based on
  - attributes, geometry or time extent
- Query can be done on the server or on the client-side.
- Different query... methods are available on **Layers** and **LayerViews**.
- Guide doc on Query and Filter

Layer and LayerView

- Every layer has a correponding LayerView
  - LayerView represents a layer in the view
- Client-side layers:
  - FeatureLayer (feature collection), CSVLayer, GeoJSONLayer
- Server-side layers:
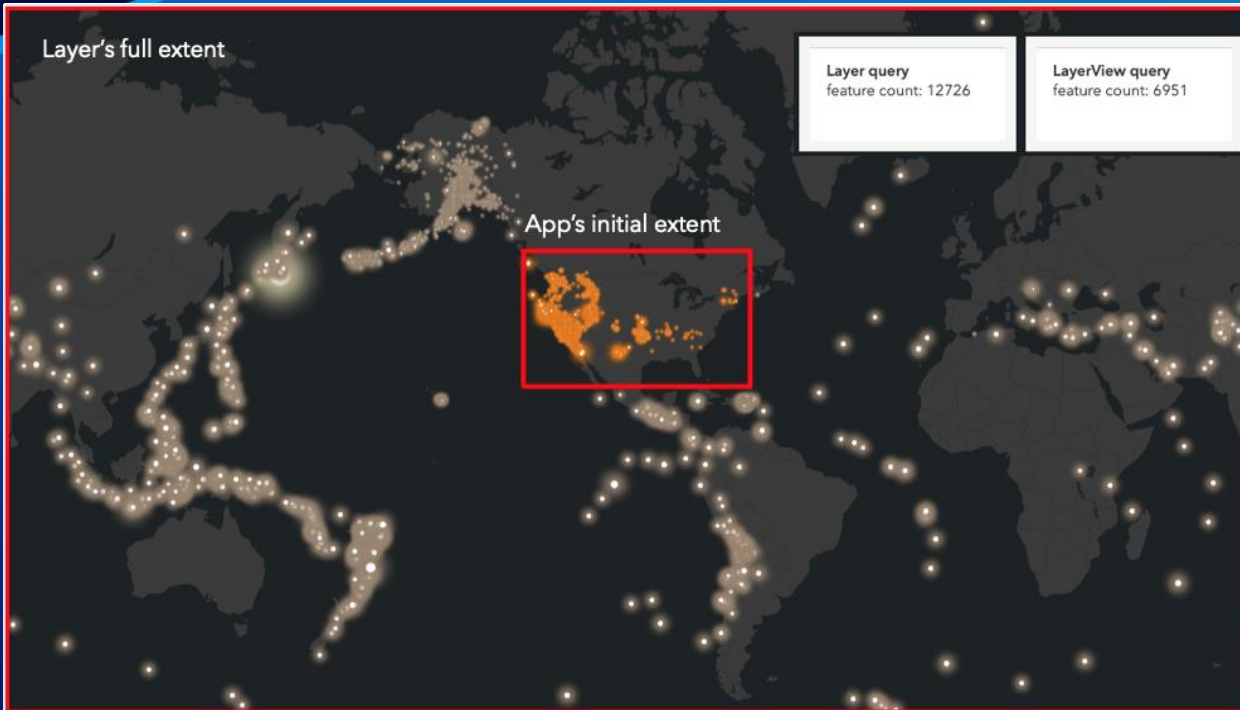 - FeatureLayer, OGCFeatureLayer, SceneLayer, StreamLayer

Server-side layers

- FeatureLayer created from a service, OGCFeatureLayer, SceneLayer, and StreamLayer
- Fetch or stream features on demand
- Server-side layer methods and properties will issue a server-side request.
  - Examples: query features, setting definitionExpression

LayerView

- [LayerView](#) represents the view for a layer after it has been added to a View.

- LayeView is responsible for rendering features in the view.

- All server and client side layer have correponding layer views.

- LayerView API is layer agnostic.

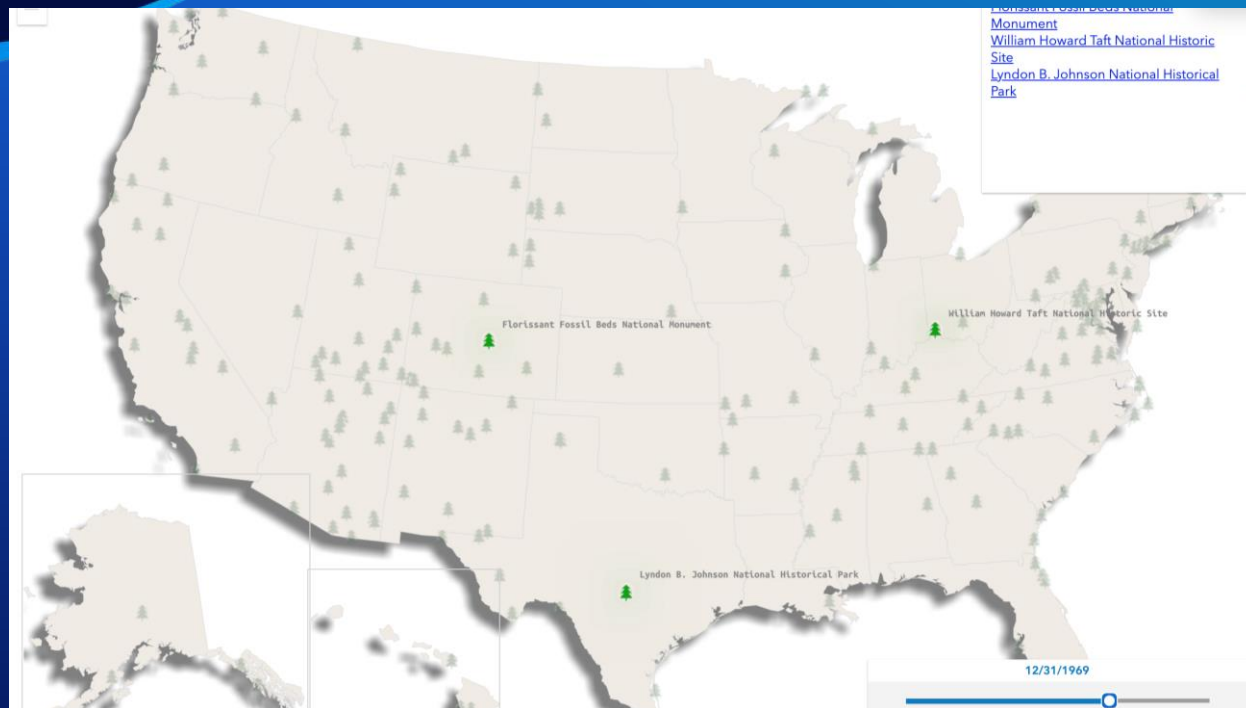- Methods and properties used against features available for drawing on the client-side.
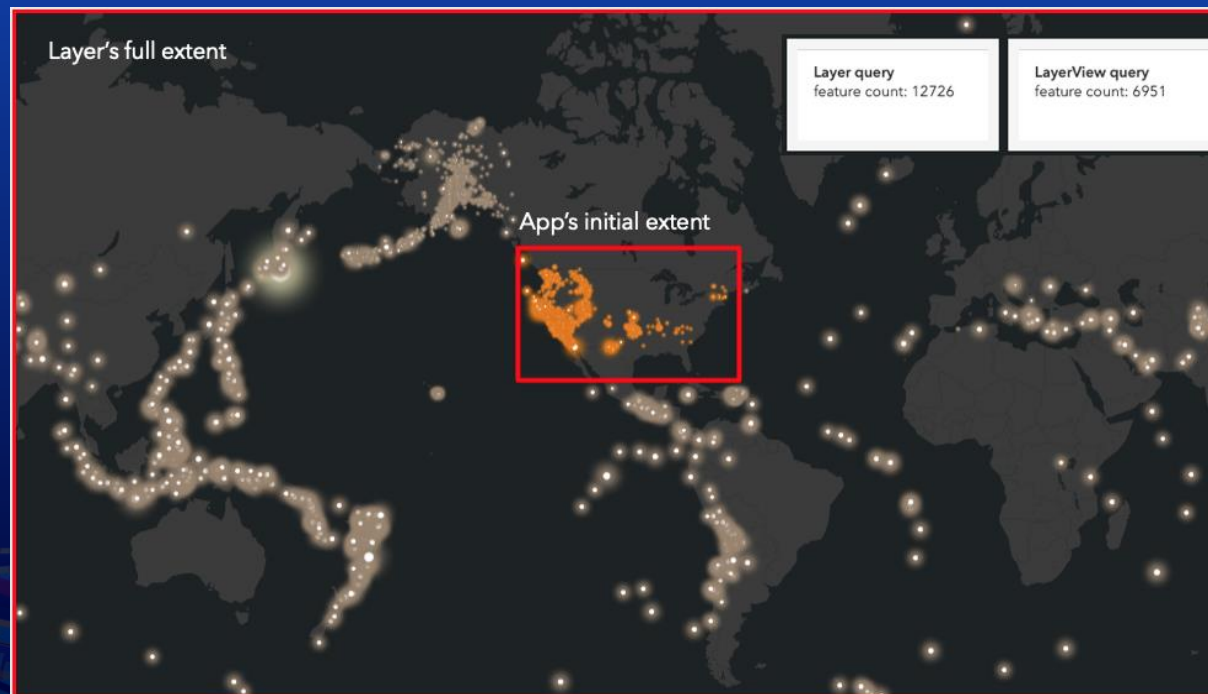
Layer vs LayerView queries

Client-side query

- (CSV|GeoJSON)Layer and FeatureLayer/FeatureCollection

- (CSV|GeoJSON|Feature|OGCFeature|Scene|Stream)LayerView

- Query methods on layer and layerView
  - queryFeatures()
  - queryFeatureCount()
  - queryObjectIds()
  - queryExtent()

CSVLayer – query features

Query tips

Should I use client-side query or server-side query?

What matters?	Server side query	Client-side query
Speed and responsiveness	**No** for server-side layers. You are making network requests, so it is slower compared to client-side queries.	**Yes** for client-side layers and layer views.
Geometry precision	**Yes.** Geometry precision is preserved. Use when it is important to get accurate results with a precise geometry.	**Yes** for client-side layers to query user provided geometries. **No** for layer views as geometries are generalized for drawing. The results can be imprecise and change as the user zooms in the map. Examples include calculating an area for selected geometries, or getting points contained in a polygon.
Must query every feature	**Yes.** Use a server-side query to make sure that the query runs against all features. Paginated queries must be done when you need to get more than max record count.	**Yes** for CSVLayer, GeoJSONLayer and client-side FeatureLayer. **No** for layerViews as the query will only run against features that are available on the client-side.

Client-side Filters

Filters and Effects

What are Filters?

- Hides features that do not match filter criteria
- Filter criteria can include spatial, aspatial, temporal conditions (or any combination)
- Client-side only. Fast.

```javascript
const layer = new FeatureLayer({ url });
const view = new MapView({
  container: "viewDiv",
  layers: [layer]
});


const featureLayerView = await view.whenLayerView(layer);

featureLayerView.filter = new FeatureFilter({
  geometry: myGeometry,     // Spatial
  timeExtent: myTimeExtent, // Temporal
  where: myWhere            // Aspatial
});
```

What are Effects?

- <u>Effects</u> are similar to Filters. Apply visual effects to *included* and *excluded* features.
- Can be used to accentuate or deemphasize feature(s).

```
const layer = new FeatureLayer({ url });
const view = new SceneView({
  container: "viewDiv",
  layers: [layer]
});

featureLayerView.effect = new FeatureEffect({
  filter: new FeatureFilter({
    where: "magnitude >= 7"
  }),
  excludedEffect: "grayscale(100%) opacity(0.5)",
  // includedEffect: "saturate(150%)"
});
```
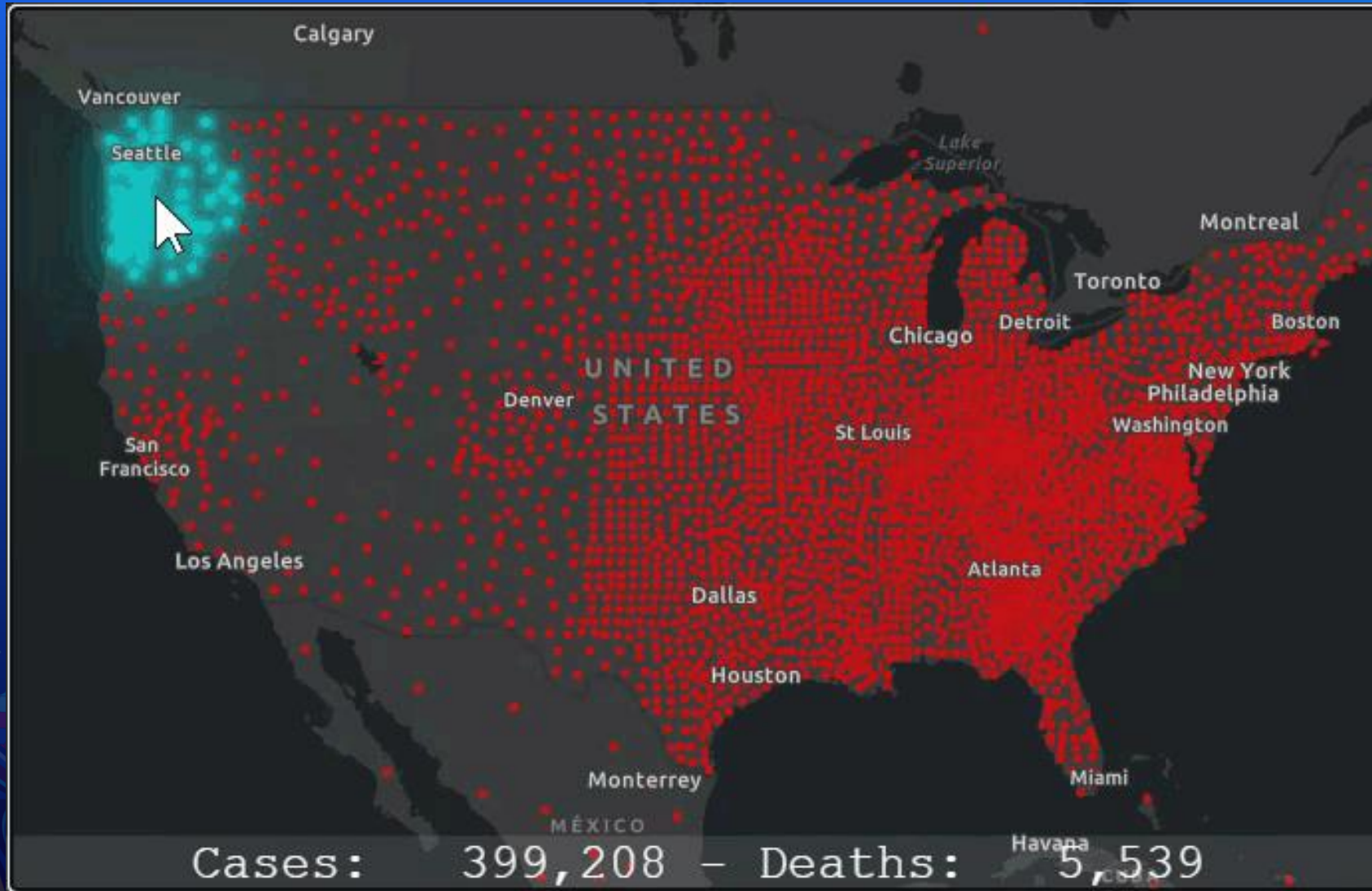
<u>CSS reference</u>

What are Effects? - Example

- Map show earthquakes from 1900 to the present day.
- Historic earthquakes are grayed-out.
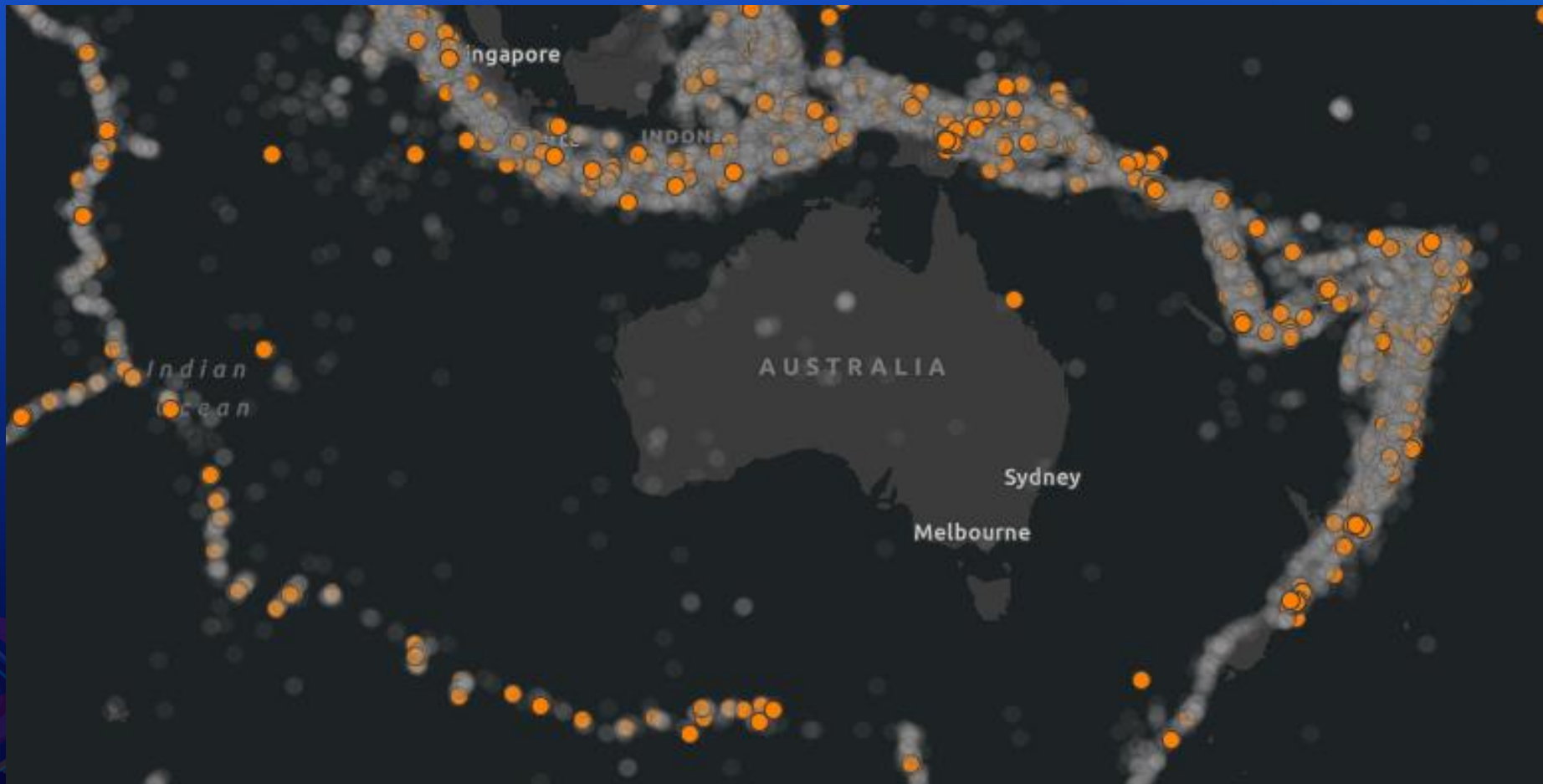
view live
source code

Filters and Effects - Demonstration

view live
source code

Client-side Analysis

Geometry Engine, Projection Engine and Geodesic Utilities

Geometry Engine

- Thirty client-side analytical methods

- Three groups
    - Compute new geometries
      e.g. buffer, clip, densify, generalize
    - Test spatial relationship
      e.g. contains, crosses, intersects
    - Advanced computations
   *e.g. geodesicArea, planarArea*

Geometry Engine – Two Flavors

- geometryEngine

```
const geometryEngine = new GeometryEngine();
const length = geometryEngine.planarLength(myPolyline, "meters");
console.log("length(m)", length);
```

- geomeryEngineAsync

```
const geometryEngine = new GeometryEngineAsync();
const length = await geometryEngine.planarLength(myPolyline, "meters");
console.log("length(m)", length);
```

Geometry Engine – Example
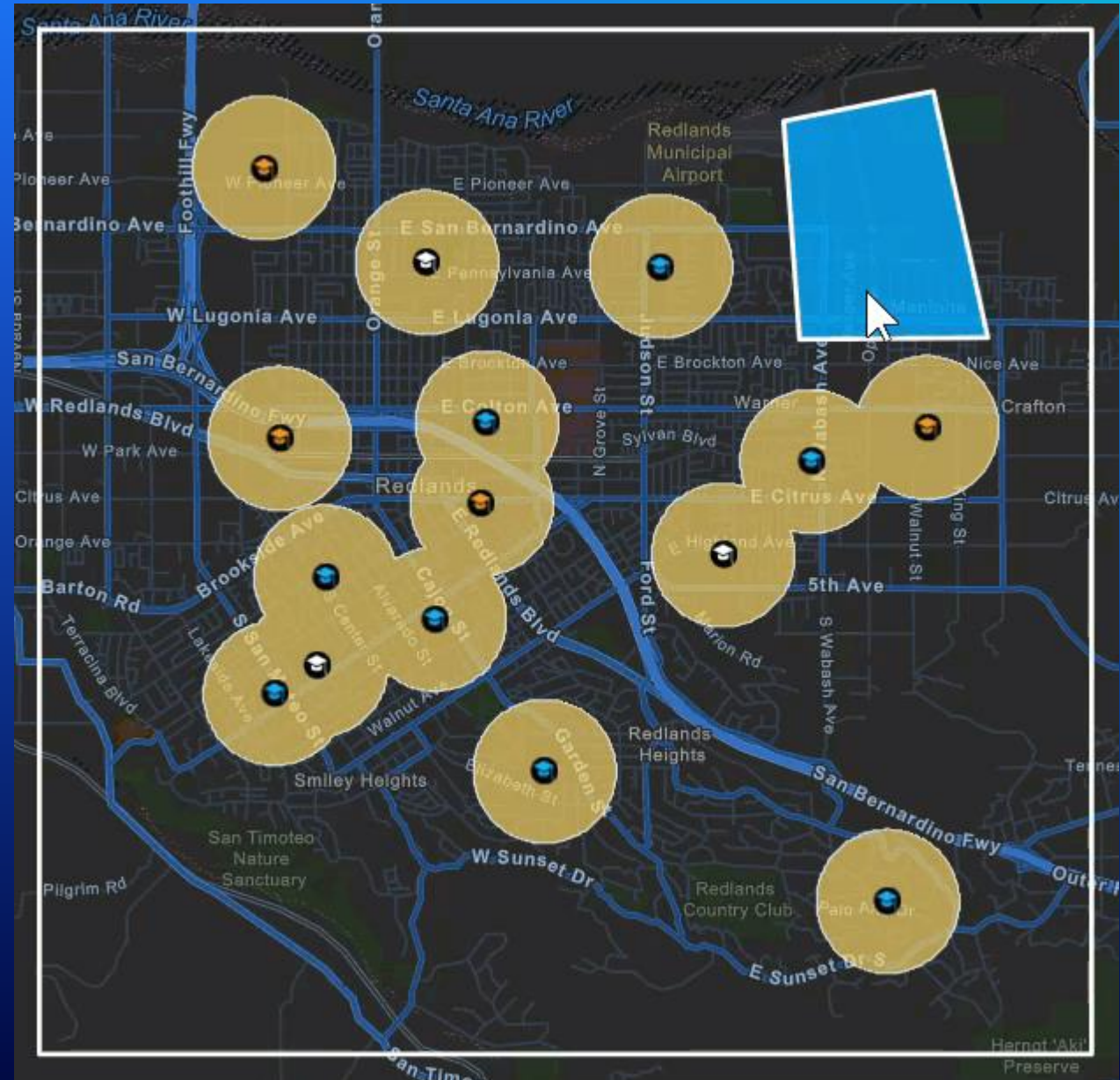
Wait, what about GeometryService?

- GeometryServer has same methods as GeometryEngine/GeometryEngineAsync
- GeometryService sends requests to a remote geometry server.
- Advantage
    - May be faster for very large or complex computations
- Disadvantages
    - May be slower due to network latency

GeometryService - Example

Find the planar length of polyline *myPolyline*.

```javascript
const parameters = new LengthsParameters({
  calculationType: "planar",
  lengthUnit: "esriSRUnit_Meter",
  polylines: [myPolyline]
});

const geometryService = new GeometryService({
  url: "http://sampleserver6.arcgisonline.com/arcgis/rest/services/" +
       "Utilities/Geometry/GeometryServer"
});

const { lengths } = await geometryService.lengths(parameters);
console.log("length(m)", lengths[0]);
```

Projection Engine

- Projection is a client-side module for projecting geometries between coordinate systems
- Based on the same code used by Runtime and Desktop
- Uses WebAssembly
- Time consuming to load

```javascript
const projection = new Projection();

if (!projection.isLoaded()) {
  await projection.load();
}

console.log("projection loaded");
```

Projection Engine - Example

Project graphics from Equal Earth Greenwich to World Fuller.
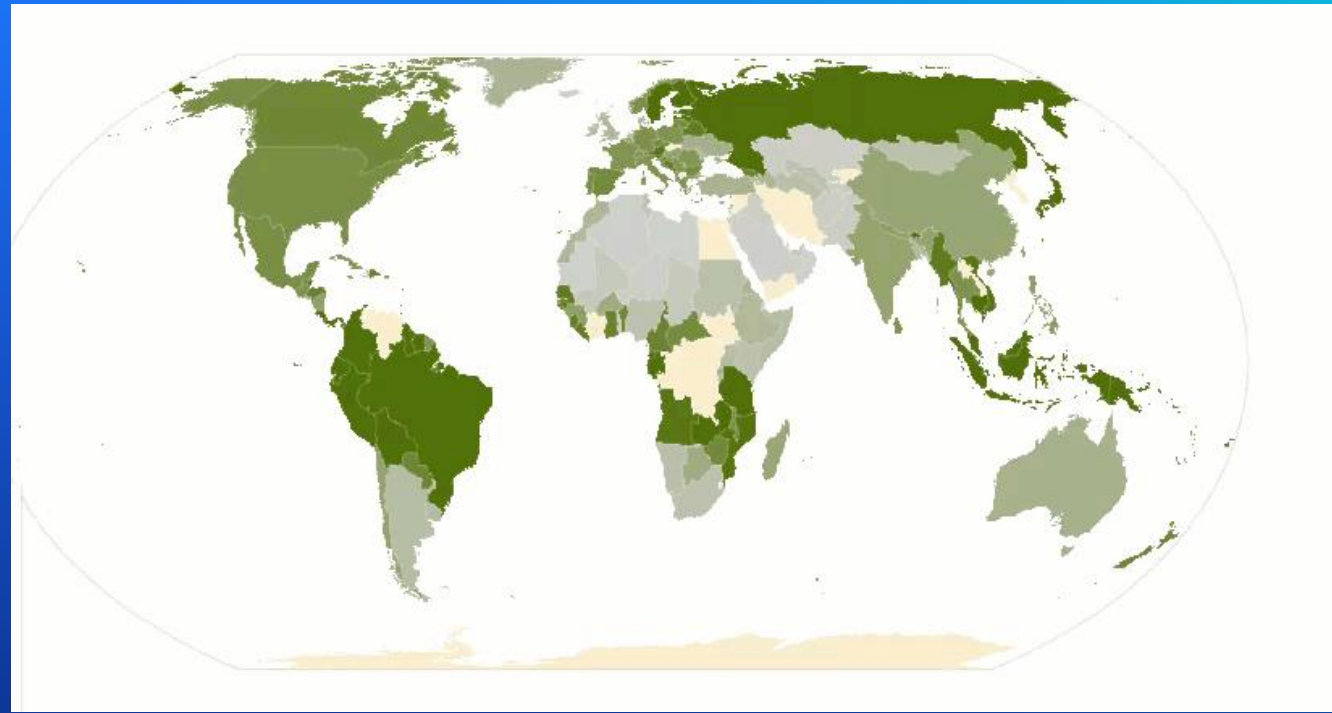
```javascript
const equalEarthGraphics = graphicsLayer.graphics;

const fullerGraphics = equalEarthGraphics.clone();

for (const graphic of fullerGraphics) {
  graphic.geometry = projection.project(graphic.geometry, {
    wkid: 54050 // World Fuller Map Projection
  });
}
```

Geodesic Utilities

- GeodesicUtils is a lightweight module for performing geodesic computations
- Some redundancy with GeometryEngine:
  - geodesicUtils.geodesicAreas() vs. geometryEngine.geodesicArea()
  - geodesicUtils.geodesicDensify() vs. geometryEngine.geodesicDensify()
  - geodesicUtils.geodesicLengths() vs. geometryEngine.geodesicLength()
- Works on any supported geographic coordinate system including 70+ non-terrestrial system (e.g. Mars and Moon)

Geodesic Utilities – Example

What is the area of the Bermuda Triangle?

```javascript
const MIAMI = [-80.208889, 25.775278];   // Florida
const HAMILTON = [-64.782, 32.293];       // Bermuda
const SANJUAN = [-66.063889, 18.406389]; // Puerto Rico

const polygon = new Polygon({
  rings: [[ MIAMI, HAMILTON, SANJUAN, MIAMI ]]
});

const [area] = geodesicUtils.geodesicAreas([polygon], "square-kilometers");

const formatter = new Intl.NumberFormat("en-us", {
  minimumFractionDigits: 0,
  maximumFractionDigits: 0
});
console.log(`Area: ${formatter.format(area)} km²`); // Area: 1,150,498 km²
```

Geodesic Utilities – Another Example

How far is the [Curiosity](#) rover (active) from [Opportunity](#) (inactive) on [Mars](#)?

```javascript
const CURIOSITY = [137.2, -4.6];
const OPPPORTUNITY = [354.4734, -1.9462];

const directLine = new Polyline({
  paths: [[CURIOSITY, OPPPORTUNITY]],
  spatialReference: { wkid: 104905 } // Mars 2000 Coordinate System
});


const [distance] = geodesicUtils.geodesicLengths([directLine], "kilometers");
const densifiedLine = geodesicUtils.geodesicDensify(directLine, 100);


const formatter = new Intl.NumberFormat("en-us", {
  minimumFractionDigits: 0,
  maximumFractionDigits: 0
});
console.log(`Distance: ${formatter.format(distance)}km`); // Distance: 8,428km
```
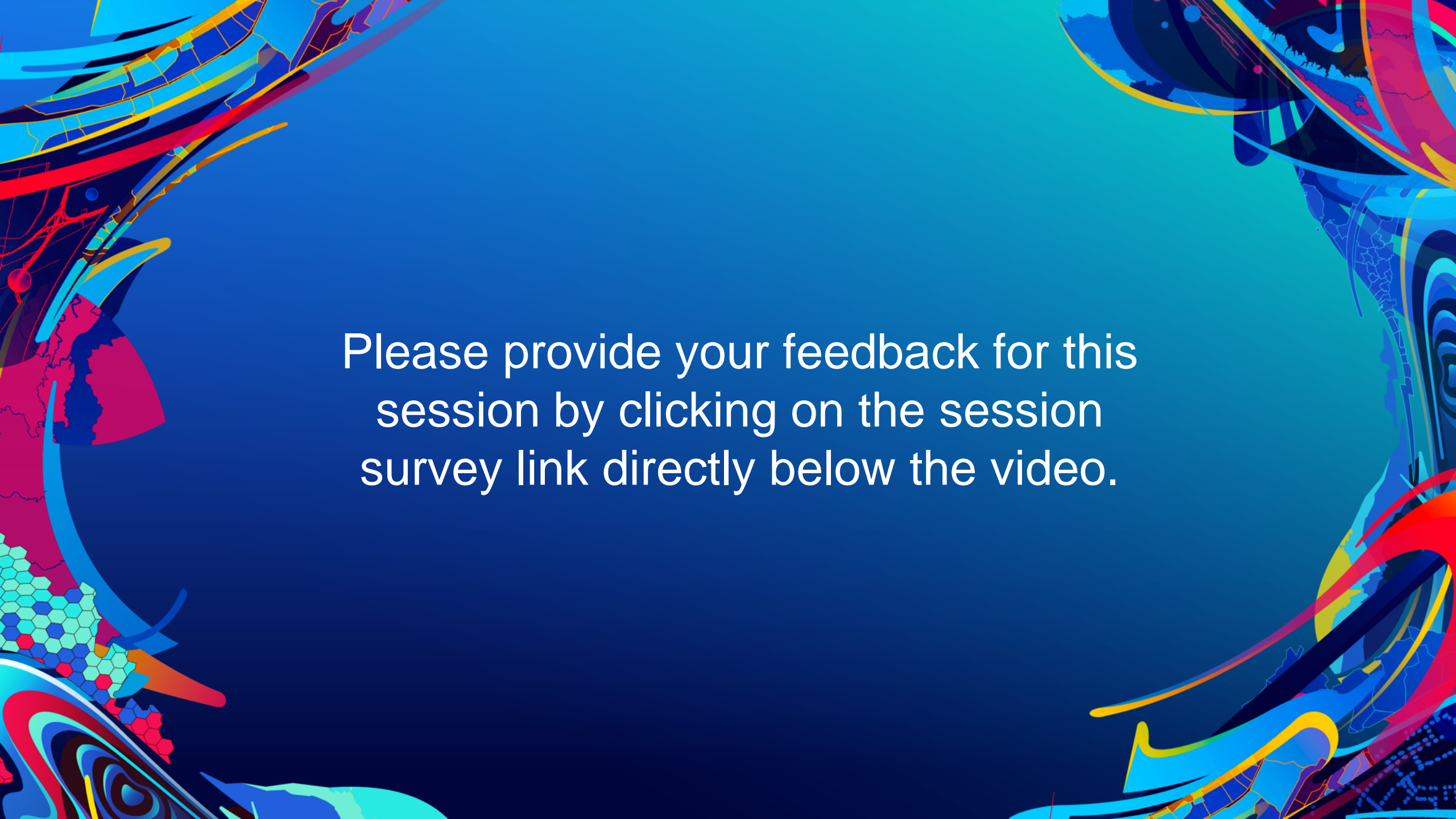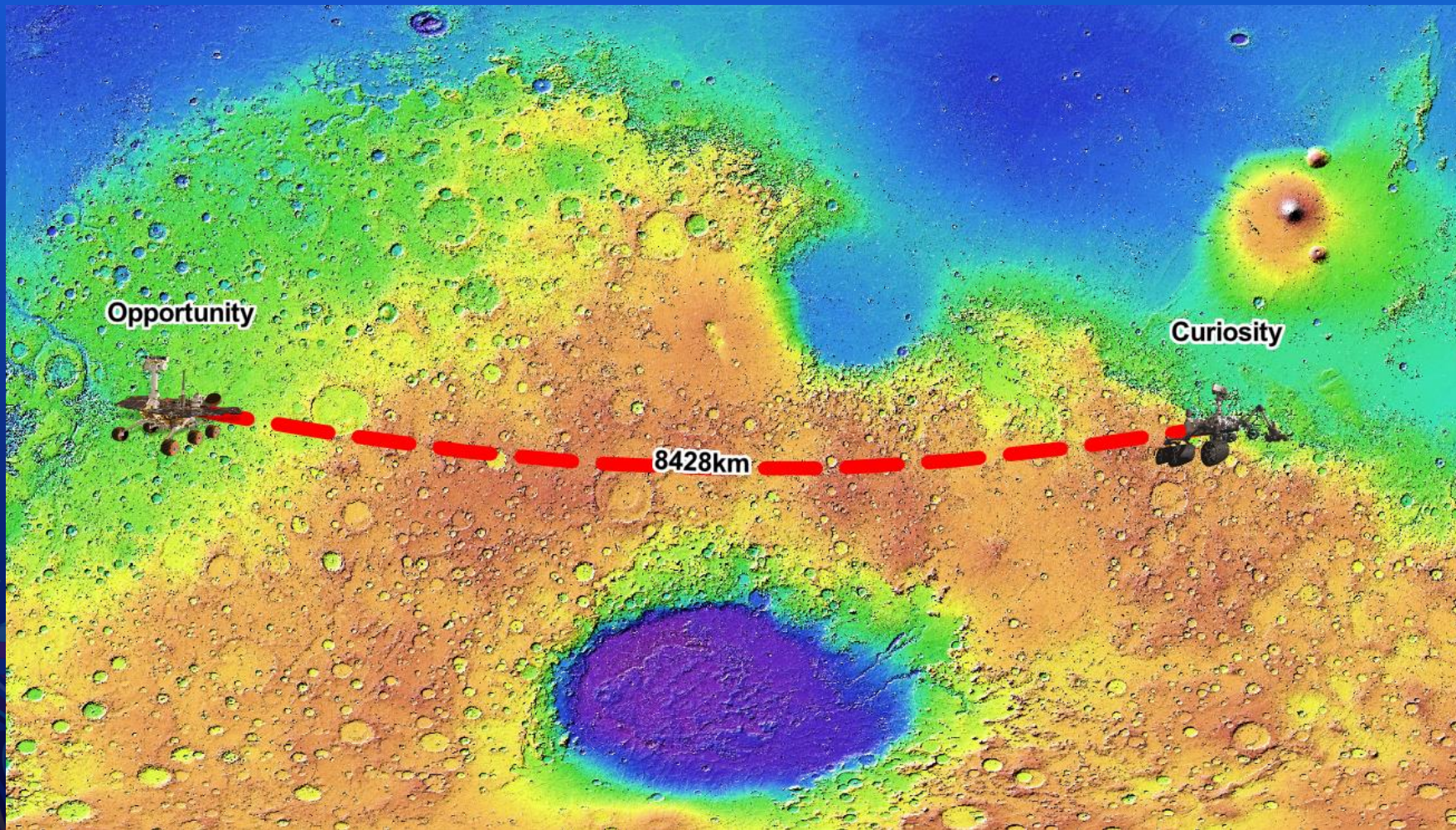
Geodesic Utilities

Finding the distances, on Mars, between Opportunity and Curiosity

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