Interactive Client-Side Mapping with the ArcGIS API for JavaScript

Undral Batsukh [ubatshukh@esri.com] | Richie Carmichael [@kiwiRichie]

`https://git.io/JfgCI (printer friendly, pdf)`

Agenda

- Client-side Layers
  - FeatureLayer, CSVLayer, GeoJSONLayer
- Query
  - Layer vs LayerView
  - Client-side query
- Filters and Effects
  - Adjusting client-side visuals
- Geometry Engine, Projection Engine and Geodesic Utils
  - Client-side analysis

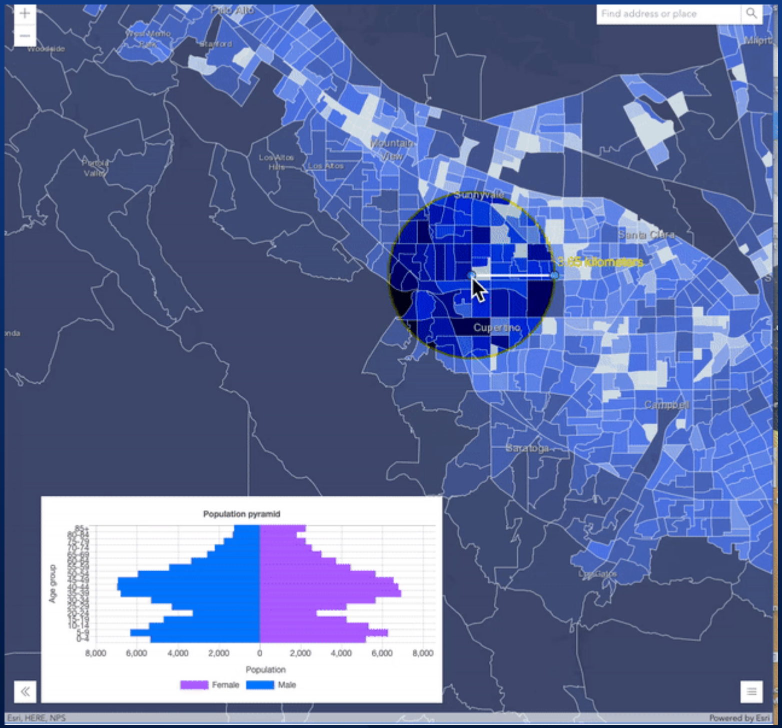
Client-side Layers

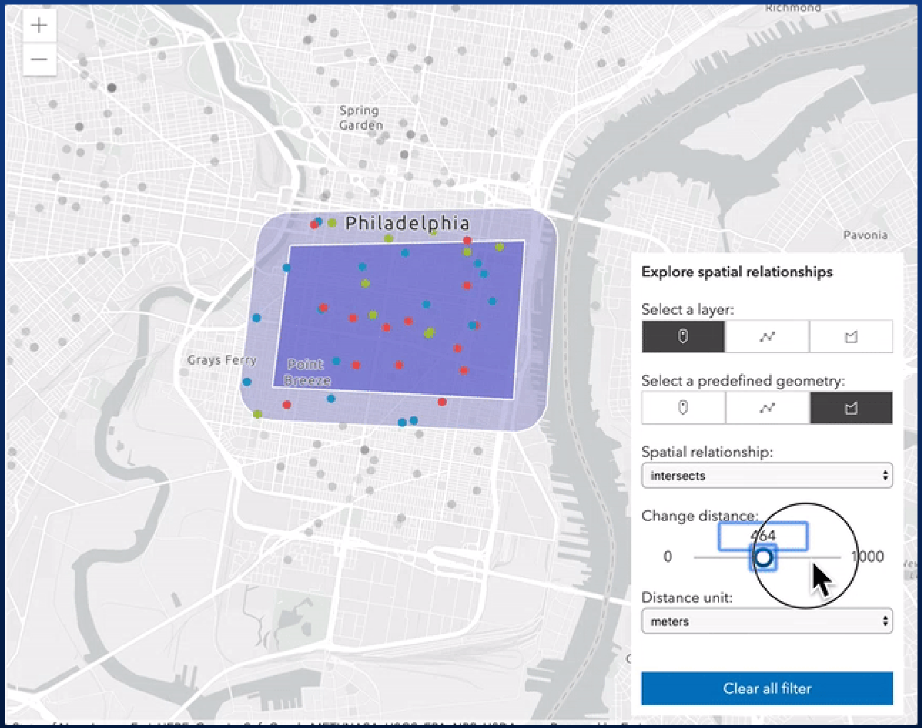
Client-side Layers

- 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

```
const 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" },
  }
})
```

API Reference | Sample 1

CSVLayer - Tips

- X, Y coordinates must be in WGS84 in csv file.
- Specify the layer's spatial reference to improve the performance.
- Not supported:
  - No z-values support.
  - Cannot add, remove or update features.

CSVLayer - Tips

- Can pass data by a blob url.

```
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*

```
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" })
  ]
})
```

API Reference | Sample 1

FeatureLayer - Tips

- Supports 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 to get the updated feature collection.

Sample - add/remove graphics

GeoJSONLayer

- Add GeoJson data that comply with the RFC 7946 specification

```
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 }
});
```

API Reference | Time-aware GeoJSONLayer

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

```
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 layers tips

- Each implementation uses the client-side query engine.
- Prefer `GeoJSON` over `CSV`.
- Proper attribution using `copyright` property.

Query

Query

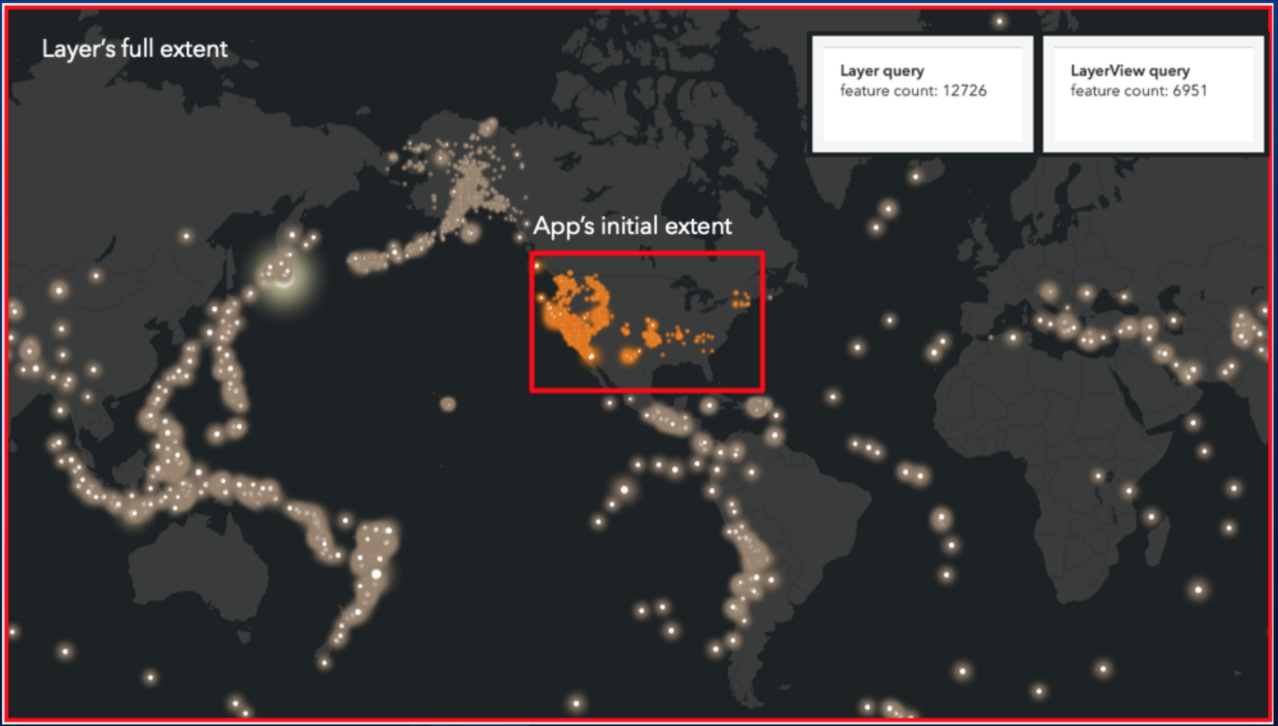
- Query expressions are used to select a subset of features and table records.
- Query can be done on the server or on the client-side.
- Different `query...` methods are available on Layers and LayerViews.
- Guide doc on Query and Filter

Layers and LayerViews

- Server-side layers
  - Fetch or stream features on demand
  - FeatureLayer created from a service, SceneLayer, and StreamLayer
  - Methods and properties for server-side layers will issue a server-side request. Examples: query features, setting definitionExpression or changing renderer.

LayerViews

- A LayerView represents the view for a single layer after it has been added to a View.
  - LayerView API is layer agnostic.
  - Methods and properties used against features available for drawing on the client-side.



Client-side query

- `(CSV|GeoJSON)Layer` and `FeatureLayer/FeatureCollection`
- `(CSV|GeoJSON|Feature|Scene|Stream)LayerView`
- Query methods on layer and layerView
  - `queryFeatures()`
  - `queryFeatureCount()`
  - `queryObjectIds()`
  - `queryExtent()`

Layer vs LayerView | Age Pyramid | 3D buildings

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.

Filters

What are filters?

- Reduce the number of features shown screen
- Can apply spatial, aspatial, temporal (or any combination)

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

- Client-side
  - Only applied to features currently downloaded
  - Fast.
- FeatureFilter has the same properties as Query

Spatial Filter

Only show buildings within 10 miles of the mouse cursor

```
mapView.on("pointer-move", (event) => {
  buildingLayerView.filter = new FeatureFilter({
    geometry: mapView.toMap({
      event.x,
      event.y
    }),
    distance: 10,
    units: "miles"
  })
});
```

Aspatial Filter

Only show earthquakes with a magnitude greater than 7

```
featureLayerView.filter = new FeatureFilter({
  where: "magnitude >= 7"
});
```

Temporal Filter

Only show earthquakes that occured between 2000 and 2007

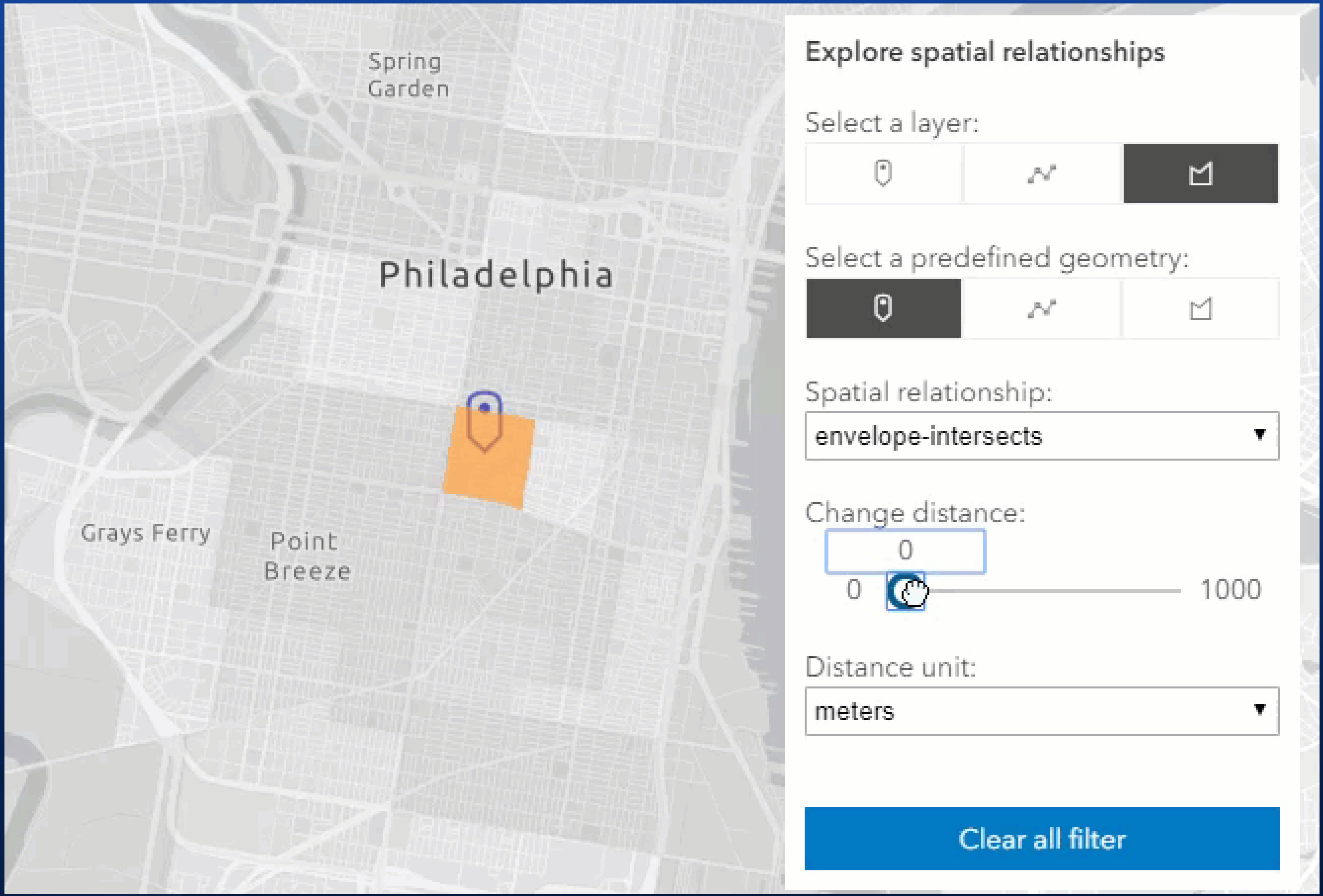
```
featureLayerView.filter = new FeatureFilter({
  timeExtent: new TimeExtent({
    start: new Date(2000, 0, 1),
    end: new Date(2007, 0, 1)
  })
});
```

Demonstration - filters

Effects

What are effects?

- Visual effects applied to included or excluded features.

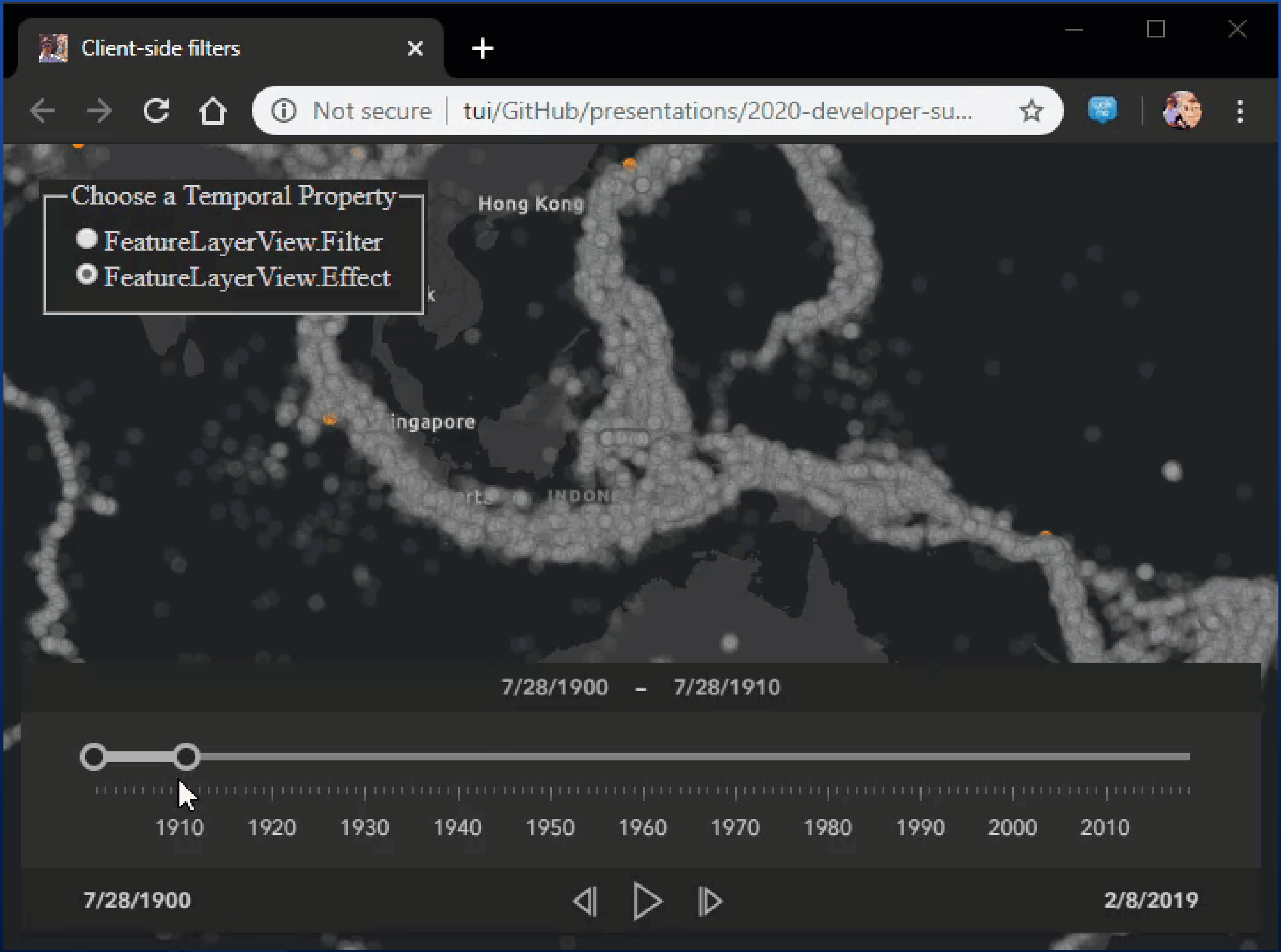


Effect - Snippet

Show earthquakes with a magnitude of 7 or greater as faint shadows

```
// Show quakes less than 7 magnitude as faint shadows.
featureLayerView.effect = new FeatureEffect({
  filter: new FeatureFilter({
    where: "magnitude >= 7"
  }),
  excludedEffect: "grayscale(100%) opacity(0.5)",
  // includedEffect: "saturate(150%)"
});
```

Demonstration - century of quakes



Supported Effects

```css
/* effect(default-value) */
brightness(0.4);
contrast(200%);
grayscale(50%);
hue-rotate(90deg);
invert(75%);
opacity(25%);
saturate(30%);
sepia(60%);
```

CSS reference

Geometry Engine

One Engine - Thirty Methods

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

One Engine - Two Flavors

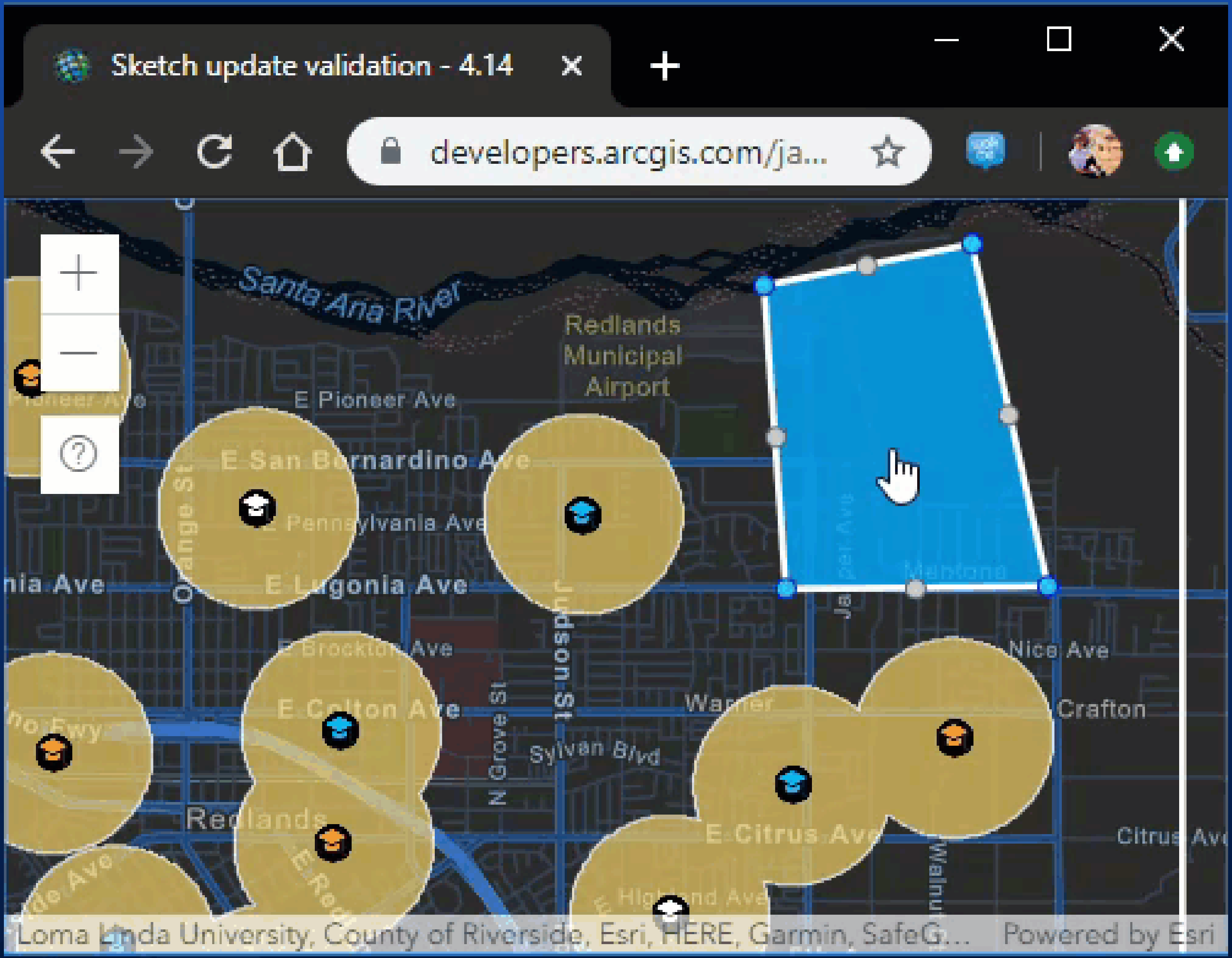
- geometryEngine

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

- geometryEngineAsync

```
const geometryEngine = new GeometryEngineAsync();
geometryEngine.planarLength(myPolyline, "meters").then((length) => {
  console.log("length(m)", length);
});
```

Demonstration - Sketch Validation



Wait, what about GeometryService?

- Same methods as GeometryEngine/GeometryEngineAsync
- Sends request to remote server for computation
- **Pro:**
  - May be faster for complex computations
- **Con:**
  - May be slower due to network latency

GeometryService - Snippet

```
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"
});

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

Projection Engine

What is Projection Engine

- Client-side module for projecting geometries between coordinate systems
- Based on the engine used by Runtime
- Uses WebAssembly
    - Not support by Internet Explorer (see caniuse)

Loading the Projection Engine

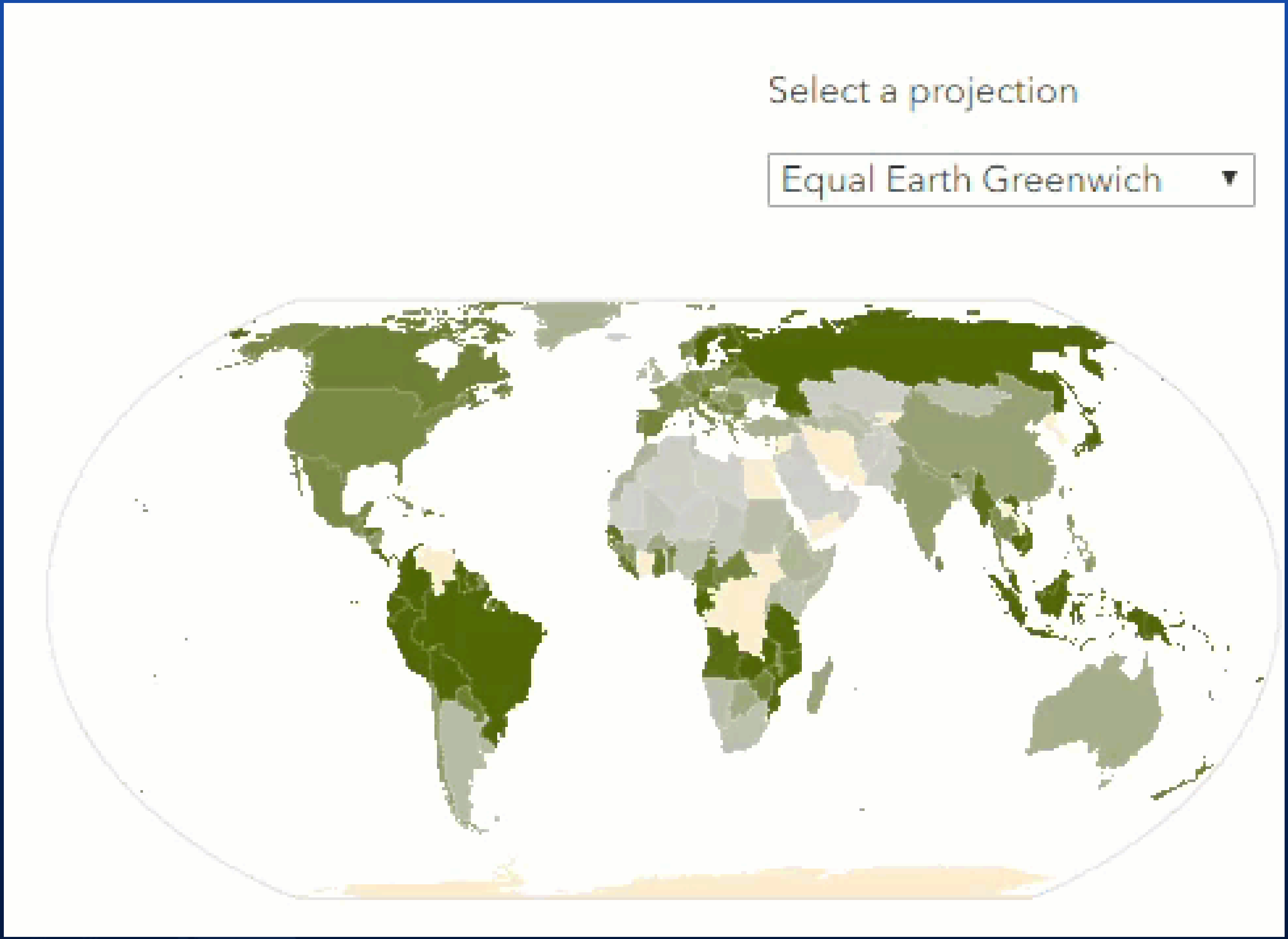
```
const projection = new Projection();

if (!projection.isSupported()) {
  console.error("projection is not supported");
  return;
}

if (projection.isLoaded()) {
  console.log("projection already loaded");
  return;
}

projection.load().then(() => {
  console.log("projection loaded");
});
```

Demonstration - Projection



Geodesic Utils

What is Geodesic Utils?

- Lightweight module for performing geodesic computations
- Some redundancy with GeometryEngine (GE)
  - GU.geodesicAreas() vs. GE.geodesicArea()
  - GU.geodesicDensify() vs. GE.geodesicDensify()
  - GU.geodesicLengths() vs. GE.geodesicLength()
- Works on any supported geographic coordinate system include 70+ non-terrestrial systems (e.g. Mars and Moon)

Geodesic Utils - Example

What is the area of the Bermuda Triangle?

```
const MIAMI    = { lat: 25.775278, lon: -80.208889 };  // Florida
const HAMILTON = { lat: 32.293, lon: -64.782 };         // Bermuda
const SANJUAN  = { lat: 18.406389, lon:  -66.063889 }; // Puerto Rico
const polygon = new Polygon({
  rings: [[
    [MIAMI.lon, MIAMI.lat],
    [HAMILTON.lon, HAMILTON.lat],
    [SANJUAN.lon, SANJUAN.lat],
    [MIAMI.lon, MIAMI.lat]
  ]]
});
const areas = geodesicUtils.geodesicAreas([polygon], "square-kilometers");
const area = Math.round(areas[0]);
console.log(`Area: ${area} km²`); // Area: 1150498 km²
```

Geodesic Utils - Mars Rovers

How far is Curiosity (active) from Opportunity on Mars?

```
const polyline = new Polyline({
  spatialReference: {
    wkid: 104905 // Mars 2000 Coordinate System
  }
});
polyline.addPath([ curiosity.geometry, opportunity.geometry ]);

const distance  = geodesicUtils.geodesicLengths([polyline], "kilometers")[0];
const densified = geodesicUtils.geodesicDensify(polyline, 100);

const path = new Graphic({
  attributes: {
    oid: 1,
    distance
  },
  geometry: densified
});
```

Please Share Your Feedback in the App

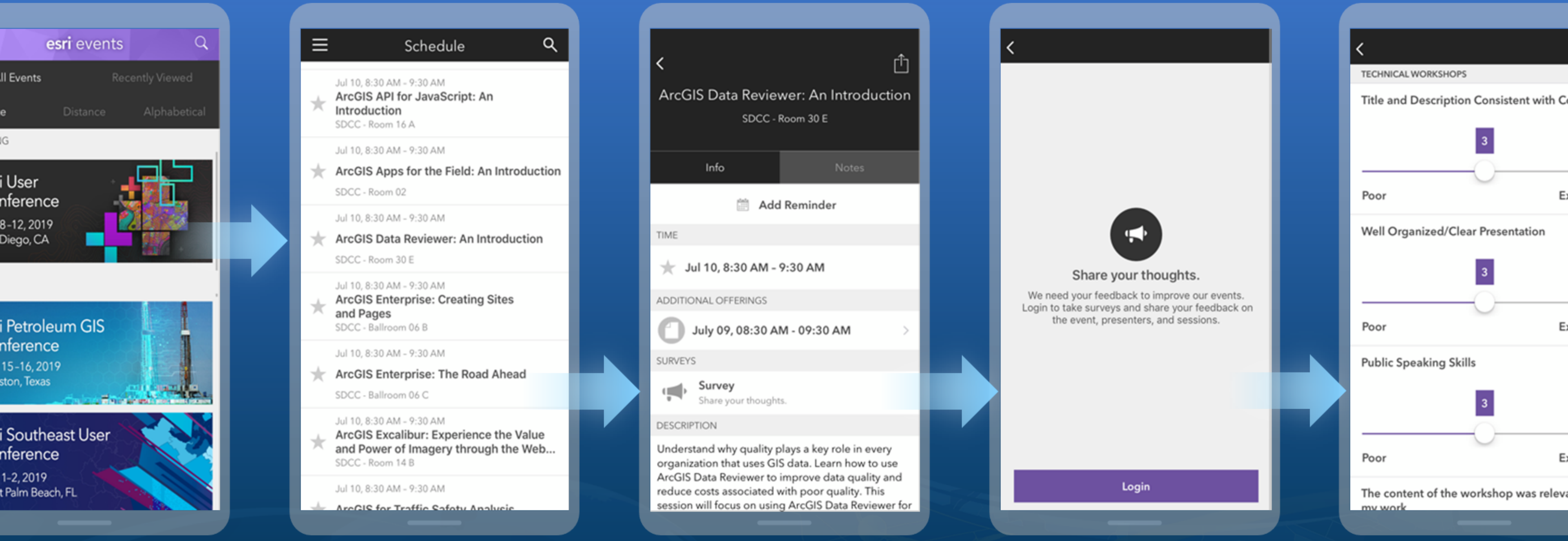
ownload the Esri
ents app and find
your event

Select the session
you attended

Scroll down to
“Survey”

Log in to access the
survey

Complete the sur
and select “Subr



esri

THE
SCIENCE
OF
WHERE

ArcGIS API for JavaScript | 2020 DevSummit Technical Sessions

http://esriurl.com/ds2020jsblog

- 28 videos focused on developing with the JS API
- Recent/relevant content (May 2020)
- All tech sessions from 2020 Esri Developer Summit are freely available on YouTube!

ArcGIS Blog

ArcGIS API for JavaScript

DevSummit 2020 – ArcGIS API for JavaScript Recordings Available!

Technical session recordings from this year's (virtual) Developer Summit are now available! This blog serves as a virtual tour for the ArcGIS API for JavaScript sessions this year.

Each year, we carefully design a technical session plan to showcase the latest capabilities of the API and best practices for building state-of-the-art, powerful, elegant, and meaningful web apps. Sessions are geared towards all levels of experience, from novice software