Assignment 6
50 Points
Time Sliders with ArcGIS JavaScript API and Leaflet.JS

The first section of this assignment will walk you through the steps of working with time-enabled data from Esri. Normally, you would be creating data in ArcGIS Pro or ArcMap desktop, setting the time properties for the map layers, and then publishing those data as a Feature Layer or feature service (to either an ArcGIS for Server instance or ArcGIS Online). In the interest of time (and server space), we will be skipping those steps initially and working with an existing data layer, global earthquakes.

In your head section, you load in Esri's scripts and set the styles. Make SURE to replace any slant quotes with straight quotes if you copy-paste this:

```
<!DOCTYPE html>
<html>

  <head>
    <meta charset="utf-8" />
    <meta name="viewport"
      content="initial-scale=1,maximum-scale=1,user-scalable=no" />

    <title>Lab 6</title>
   <link rel="stylesheet"href="https://js.arcgis.com/4.18/esri/themes/light/main.css" />
   <script src="https://js.arcgis.com/4.18/"></script>

    <style>
      html,
      body,
      #viewDiv {
        padding: 0;
        margin: 0;
        height: 100%;
        width: 100%;
      }

      #timeSlider {
        position: absolute;
        left: 5%;
        right: 5%;
        bottom: 20px;
      }

    </style>
    <script>
```

This final tag on the above section begins the "script" that will fire all the functions and layering for your map. Virtually all the code will go within the script tags. The script should begin with the dojo require section – this loads in the resources for the widgets we'll use. Here you will load in the map and map view features, as well as the feature layer and legend widgets you've already used in previous examples. But notice that "Time Slider" also appears here (a feature we haven't used before). Please add this section *after* the script tag, but before you add anything else:

```
require([
      "esri/Map",
      "esri/views/MapView",
      "esri/layers/FeatureLayer",
      "esri/widgets/TimeSlider",
      "esri/widgets/Legend"
    ], function (Map, MapView, FeatureLayer, TimeSlider, Legend) {
```

Notice a few things here:
1) The require section begins with a (. That means at the end of your script section, before the </script> tag to close, you will need to end with a matching) to bind it all together.

2) Likewise, your functions list opens with a { which means that you'll also need to close with a matching } when you're done with the script for the map that fires functions. Taken together, this explains why the code usually has a concluding line that looks something like this: **});**

3) Also, again, the *order* in the list of resources matches the *order* in functions list that follows it. This is not only good practice, it's also often necessary to make the page to function properly.

This list also tells us something about the expectations of the following code we'll be writing. We'll want to define a map, a map view, a feature layer, a time slider, and a legend. However, just because we'll be defining/adding all those things in order within the list of declared functions at the top doesn't mean we'll be adding them to the JS code *in that order*. Rather, the map feature expects to know what layers will be loaded in, so the feature layers have to be defined first; the map view expects to know the map's properties, so the map variable has to come before the view; the legend expects to know the properties of the map view before it can be fired, so the view has to come before the legend, et cetera…

So, we will begin by defining the thing that has to come first – the map layer itself. Unlike the previous ArcGIS Maps SDK for JavaScript exercise, we aren't using a separate URL variable. You can link directly to it in the properties of the layer, too. Here, we are adding just one layer first. This item has time properties that are recognized by Esri (a declared date field), and it is a Feature Layer:

```
const layer = new FeatureLayer({
url:
"https://services9.arcgis.com/RHVPKKiFTONKtxq3/ArcGIS/rest/services/USGS_Seismic_Data_
v1/FeatureServer/0"
      });
```

Next, we can add the map that will *use* that feature layer:

```
const map = new Map({
        basemap: "hybrid",
        layers: [layer] //if you add more layers, separate by a comma
      });
```

The map view will be next:

```
const view = new MapView({
        map: map,
        container: "viewDiv",
        zoom: 2, //zoomed almost all the way out to see the globe
        center: [-150, 0] //set this somewhere in the middle of the Pacific
      });
```

Now we have the map starting position set, we have a base map, and we have a layer drawn onto the map. It is time to begin adding map elements like the Legend as well as interactive functions, in this case, the time slider -

```
const timeSlider = new TimeSlider({
    container: "timeSlider",
    view: view,
    timeVisible: true, // show the time stamps on the timeslider
    loop: true
   });

view.whenLayerView(layer).then(function (lv) {
        // this just rounds up the time extent to full hours
 timeSlider.fullTimeExtent = layer.timeInfo.fullTimeExtent.expandTo("hours");
      });

const legend = new Legend({
        view: view
      });

view.ui.add(legend, "top-left"); //you can position the legend here

});
 </script>
  </head>
```

Now we have the entire script section completed for the basics of what we want. It's still not done! This is just the head section of the HTML document where we've preemptively loaded in all the scripts for functionality. We still need to give the map a place on the physical page the user sees. So, in the "body" section, we'll put in an HTML division, essentially the window where our map will live. We will also create a separate div for the time slider element. Notice that these div items link up with their corresponding names in the style section on page 1.

```
  <body>
    <div id="viewDiv"></div>
    <div id="timeSlider"></div>
  </body>

</html>
```

At this point, the basics of the page are in place. But let's experiment with adding another interactive element to the page: a pop up for the earthquake data.

In the ArcGIS Maps SDK for JavaScript API, pop ups do *not* need to be declared in the top/require section if you're using them only to derive data from existing feature layers. However, you will want to modify the feature layer properties anyway in order to make them work properly. We return to the feature layer variable to make some additions (note the new comma after the URL):

```
const layer = new FeatureLayer({
url:
"https://services9.arcgis.com/RHVPKKiFTONKtxq3/ArcGIS/rest/services/USGS_Seismic_Data_
v1/FeatureServer/0",
       outFields: ["depth", "mag"], // used to specify what attribute fields to query
       popupTemplate: popupTemplate
       });
```

Notice that we are pulling out two fields from the dataset (and keeping them in quotes!), and furthermore the feature layer property "popupTemplate" is set to a variable called "popupTemplate." We haven't defined that yet! So ***before*** the layer variable (but after the require section), we'll need to do that, much like you did with renderers in a previous exercise.

So add this section to the location I've just described:

```
const popupTemplate = {
          // autocasts as new PopupTemplate()
          title: "this event took place at {depth} feet deep",
          content: [
            {
              type: "fields",
              fieldInfos: [
                {
                  fieldName: "mag",
                  label: "Magnitude",
                  format: {
                    places: 0,
                    digitSeparator: true
                  }
                }]}]};
```

I have simplified this substantially, so there's only one field being populated into the table of data (the magnitude of the earthquake) that pops up. You could use commas here to extend this template further and include additional attributes for each point on the map, when clicked. Notice also that there is a pop up TITLE field. It draws from the attributes as well. In this case, I've got it set to {depth} so that it pulls the recorded depth of the quake and populates it into the pop up. Now you have everything you need for a functional, time-enabled map with pop ups (that's 10 points already!)

On your own, complete the following steps to finish your assignment:

1) change the pop-up template so that **two more attributes** from the earthquake layer are loaded into the window when you click on a point [look at the feature layer URL for ideas] (20 points)

2) change the **pop-up template title** in any way you'd like, as long as it dynamically loads in an attribute [e.g. "depth"] as part of the title (10 points)

3) change out the **base map** with a different option [consult earlier exercises/ArcGIS help if needed] (10 points)

Upload your finished work to your GitHub pages account and submit the link.