CSCI 3342/6312 – Tutorial/Assignment #3 **Deadline: 3:00 pm on June 9, 2020**

– In-Class Project/Quiz #3 (Separate *Container*) **Deadline: 3:00 pm on June 9, 2020**

Completion: Required **Submission: Required**

Last Name:_____ First Name:_____

This *tutorial* is to help you learn basic ***REST API*** programming to interface your webpage to an already written application available on the Internet that allows your webpage to interface with it.

1) How and what to submit?

Submit the following (upload in Blackboard to the available container) in "one" PDF document (not in docx or any other format):

i) The certification page (see next page) should be the first page, followed by
ii) your solution to the problems given on this assignment.

One way is to copy the certification page and your solution to the problems into a Word document and then save the Word document as PDF, and upload the PDF version (not docx version). Only the PDF version will be graded.

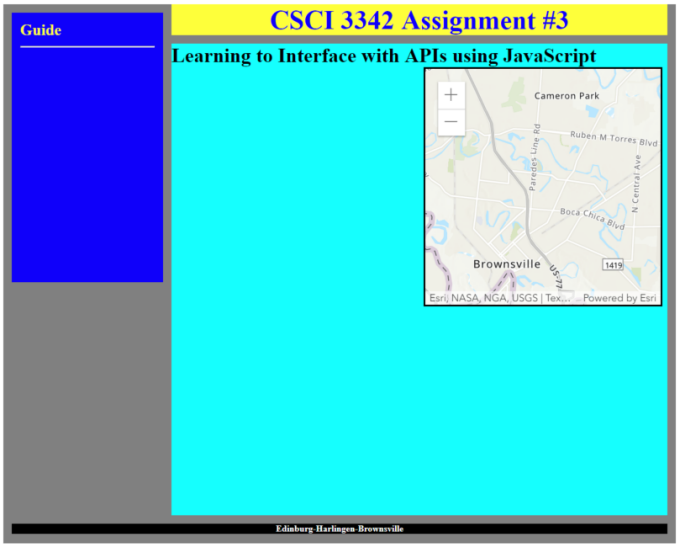
2) Only ONE upload attempt is allowed: Before submitting a document through Blackboard, you should review the document being uploaded to make sure that you are uploading the correct document (e.g. do not upload the assignment belonging to another course). To help you prevent uploading wrong documents, notes (titled "HelpOnSubmissionThroughBlackboard" on how to save & review drafts before final submission have been uploaded under Reference Material folder.

Certification Page

This page must be the first page of your uploaded document.
Your assignment will not be graded without this page (completed with your full name in the area provided) as the first page of your uploaded document.

I, _____, certify that the work I am uploading represents my own efforts, and is not copied from anyone else or any other resource (such as Internet). *Furthermore, I certify that I have not let anyone copy from my work.*

Tutorial Portion



Learning Objectives!

An ***API*** (Application Programming Interface) is a way for a software programs to talk with each other. For example, we have APIs for Youtube, Google, and Twitter. An API may require an ***API-Key*** to be attached with every request for the API entity to respond. In this tutorial we will learn how to interface our webpage with the ***MapViewer*** API of ***ArcGIS*** (Arc Geographical Information System) managed by ESRI (Environmental Systems Research Institute). As shown in the above snapshot, we will set aside a rectangular area in our webpage to view map of a location. We will be using ***REST*** (REpresentational State Transfer – using http to exchange data) API. In REST protocol, our code will send a ***stateless*** request (meaning, the current request is sent with complete information and does *not* depend on the components of previous requests), and the remote server will return relevant data to update its application's output being shown in the designated area. Below is a summary of each section in this tutorial:

Section 1: Build a basic webpage before starting on REST API.

Section 2: Designate an area on the webpage for ArcGIS's *MapViewer* REST API.

Section 3: Use the JS code provided by ArcGIS to show the map of a location in the designated area.

Section 4: Create a few fields on the webpage to accept address/latitude/longitude to show map of a specific location.

Section 5: This section is ***optional***. This is to learn how to interface a webpage with a Google's API which requires an API-Key. Since Google requires one to create an account and provide a credit card, it is up to you if you want to do this section or not.

Section 6: This section is for ***information*** only. If REST APIs are something that you want to get more into, the ESRI site provides useful tutorials for you to build your own applications. *The information provided in this section may be helpful in completing Quiz #3.*

(1) Let's start with the simplified version of the template that we have been using, represented by the following a3.html and e3.css (note that the files have been simplified to remove clutter so that we can concentrate on the API interfacing):

Initial a3.css file:

```
<!DOCTYPE html>
<html>
	<head>
		<title> CSCI 3342 Assignment #3 </title>
		<link rel="stylesheet" type="text/css" href="e3b.css" />
	</head>
	<body>
		<div id="canvas">
			<div id="guide">
				<h3>Guide</h3>
				<hr>
			</div>
			<div id="header">
				<h1>CSCI 3342 Assignment #3</h1>
			</div>
			<div id="body">
				<h2 id="head1">Learning to Interface with APIs using JavaScript</h2>
			</div>
			<div id="footer">
				<h6>Edinburg-Harlingen-Brownsville</h6>
			</div>
		</div>
	</body>
</html>
```

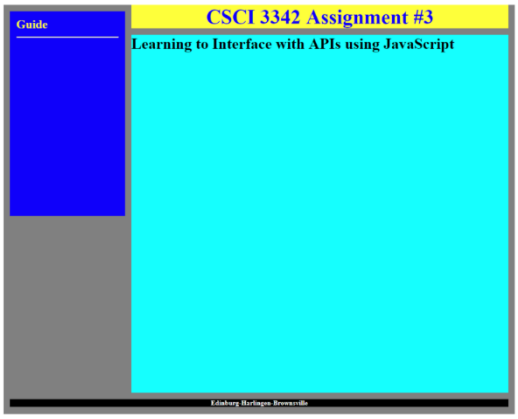
Initial e3.css file:

```
/* This is the CSS file for Assignment/Tutorial #3 */
#canvas {
	background-color: grey;
	width: 800px;
	height: 640px;
}
#guide {
	background-color: blue;
	color: yellow;
	width:160px;
	height:300px;
	margin-left: 10px;
	margin-top:10px;
	float: left;
	padding:10px;
}
#header {
```

```
        background-color: yellow;
        color: blue;
        width: 590px;
        margin-left: 200px;
        margin-top: 10px;
        text-align: center;
}
#body {
        background-color: cyan;
        color: black;
        width: 590px;
        height: 560px;
        margin-left: 200px;
        margin-top: 10px;
}

#footer {
        background-color: black;
        color: white;
        margin-left: 10px;
        margin-right: 10px;
        margin-top:10px;
        text-align: center;
}
h1, h2, h3, h4, h5, h6 {
        margin: 0px;
}
```

Run file:///home/pi/a2.html in RPi's browser. The output should look like the following:



(2) ArcGIS provides an API (without the need to have an API-Key... at least as of June 2019) to access its map application. We would like to show a map-inset of an area within #body DIV. Let's create a sub-DIV of size 280px-by-280px within #body DIV aligned with the right margin. Modify a3.html as follows:

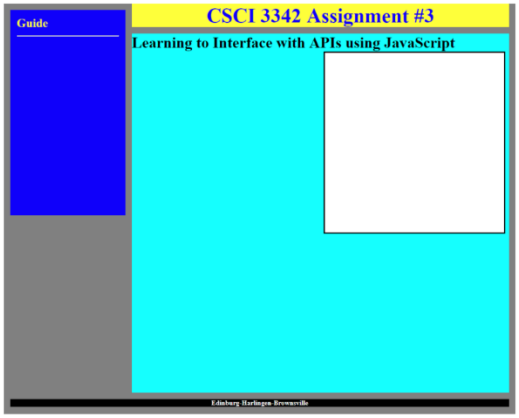
```
<div id="body">
```

```
        <h2 id="head1">Learning to Interface with APIs using JavaScript</h2>
        <div id="viewMapDiv">
        </div>
</div>
```

Add the style for ViewMapDiv in e3.css:

```
#viewMapDiv {
        border: solid 2px black;
        background-color: white;
        width: 280px;
        height: 280px;
        margin-left: 300px;
}
```

Refresh the browser. The webpage should look as follows:



(3) We need to point to the ArcGIS API by adding the following lines in <head> tag of a3.html:

```
        <meta charset="utf-8">
        <meta name="viewport" content="initial-scale=1, maximum-scale=1, user-scalable=no">
        <link rel="stylesheet" href="https://js.arcgis.com/4.11/esri/themes/light/main.css">
        <script src="https://js.arcgis.com/4.11/"></script>
```

Place the following right before </body> tag to call the JS code to be defined shortly:

```
        <script src="script3.js"></script>
        <script>initMap();</script>
```

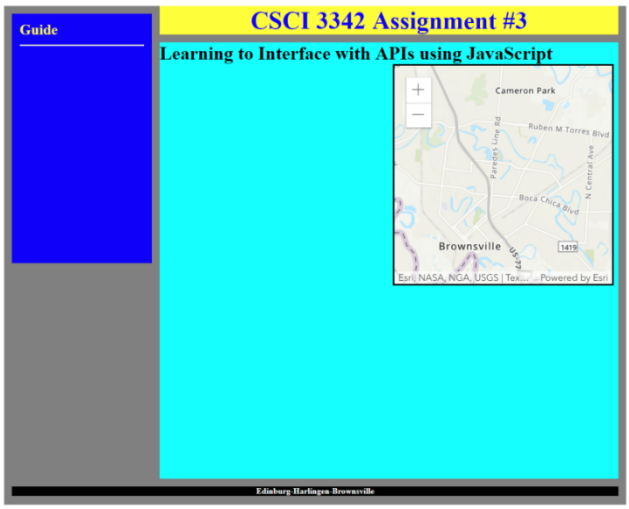
ArcGIS provides "local" JS code for our application to use in order to manipulate the map. For example, container (which DIV to the show the map in), location (longitude and latitude), zoom factor, etc. Below is the code that needs to be present locally under script3.js. It is important that "script3.js" (local JS code) is referenced after ArcGIS link lines in <head> tag because it refers to the environment (variables etc.) defined in the JS code at ArcGIS. Note that the values in blue are the ones that we are interested in manipulating:

```
//JS code for Assignment #3
function initMap() {
  require([
     "esri/Map",
     "esri/views/MapView"
   ], function(Map, MapView) {

    let map = new Map({
     basemap: "topo-vector"
    });

    let view = new MapView({
     container: "viewMapDiv",
     map: map,
     center: [-97.484424,25.930307], // longitude, latitude
     zoom: 12
    });
  });
}
```

Refresh the browser. The webpage should look as follows showing Brownsville area as the starting point:



(4) Let's say that we are interested in showing the map of Santa Fe (as a starting location). Googling for longitude and latitude for "Santa Fe New Mexico" yields [-105.944183, 35.691544]. Plug in the longitude and latitude of Santa Fe in script3.js and refresh the browser. Observe that the map shown is of Santa Fe area.

However, we would like to specify the longitude and latitude of an area through input fields rather than updating the code manually. Let's create an input field to put an address in, a "Go" button, and fields to input longitude and latitude. All the text fields must have valid default values (e.g. *Brownville Texas* data). Modify a3.html as follows:

```
<div id="body">
        <h2 id="head1">Learning to Interface with APIs using JavaScript</h2>
        <input id="address" type="text" value="Brownsville Texas"/>
        <button id="go">Go</button>
        <input id="lng" type="text" value="-97.484424"/>
        <input id="lat" type="text" value="25.930307"/>
        <div id="viewMapDiv"> </div>
</div>
```

The JS code needs to be modified to read the values from longitude and latitude fields when "Go button is pressed:

```
//JS code for Assignment #3
function initMap() {
 let lng = document.getElementById("lng").value;
 let lat = document.getElementById("lat").value;
 require([
    "esri/Map",
    "esri/views/MapView"
  ], function(Map, MapView) {

  let map = new Map({
   basemap: "topo-vector"
  });

  let view = new MapView({
   container: "viewMapDiv",
   map: map,
   center: [lng,lat], // longitude, latitude
   zoom: 12
  });
 });
}

//when "Go" button is clicked, call initMap() function
document.getElementById("go").addEventListener("click", initMap);
```

Refresh the browser. The webpage should show Brownsville coordinates and map. Enter Santa Fe's coordinates (found earlier through Google) and click on Go button. Observe that Santa Fe's map is displayed.

(5) ***Optional*** - We would like to automate getting the longitude and latitude of an area and update the map accordingly. In other words, we would like to enter the address of a location, such as "Santa Fe New Mexico," and click on Go button for the underlying detail to find coordinates and display the map accordingly.

In order to find the coordinates of a given address (and vice versa) we need Geocoding API Service. We need to provide the following functionality via Java Script:

- when "Go" button is pressed:
  - -address is extracted from *address* input field
  - -Google Geocoding API is accessed to get longitude/latitude of the specified address using geocoder() function of Google.

Add the following to <head> tag in order to access Google's API:

```
<script async defer
        src="https://maps.googleapis.com/maps/api/js?key=APIKEY" type="text/javascript">
        </script>
```

Modify script3.js to extract coordinates from Google's Geocoding API *(note the modification in addEventListener which needs to point to a new function to extract coordinates when "Go" button is pressed):*

```
function initMap() {
 let lng = document.getElementById("lng").value;
 let lat = document.getElementById("lat").value;
 require([
    "esri/Map",
    "esri/views/MapView"
  ], function(Map, MapView) {

  let map = new Map({
    basemap: "topo-vector"
  });

  let view = new MapView({
    container: "viewMapDiv",
    map: map,
    center: [lng,lat], // longitude, latitude
    zoom: 12
  });
 });
}

function showLngLat(result) {
  document.getElementById('lng').value = result.geometry.location.lng();
  document.getElementById('lat').value = result.geometry.location.lat();
}

function getLngLat(callback, address) {
  geocoder = new google.maps.Geocoder();
  if (geocoder) {
    geocoder.geocode({
       'address': address
    }, function (results, status) {
```

```
        if (status == google.maps.GeocoderStatus.OK) {
          callback(results[0]);
        }
      });
    }
}

//when "Go" button is clicked, call a function to get Lng/Lat info from Google API using geocoder()
document.getElementById("go").addEventListener("click", function () {
  getLngLat(showLngLat, document.getElementById('address').value)
});
```

Refresh the browser. In the Inspect-Console window, there should be an error regarding accessing Google API since Google requires an *API Key* with each request. *Consider* the steps given below to acquire an API Key, activate it, restrict the API Key, and enable relevant services.

I. Create a gmail account (*apparently, it is needed to get an API Key... when you activate cloud services, Google provides $300 worth of free credit after which it continues to provide service at a cost if you "manually" allow Google to do so*).

II. Access https://cloud.google.com/console/google/maps-apis/overview

III. ➔ *Project* menu ➔ Create a project (e.g. CSCI3342) for which an API key is needed.

IV. Select the created project.

V. Navigation menu ➔ **APIs & Services ➔ Credentials**

VI. **Create credentials ➔ API key** ➔ Copy the created API key ➔ **Close**

VII. Restrict your API key on the *Credentials* webpage as follows:

   a. Click on the API key that you want to restrict

   b. Find the IP address of your computer running your website by Googling "my IP address." Let's say that it is 62.123.204.36.

   c. Select *IP addresses* under *Application restrictions*. Click on *ADD AN ITEM* to add the IP address of the computer running your website (Googling "my IP address" will provide you with the public IP address of your computer).

   d. Click on **Save**.

VIII. Select (check mark) your API key and click on "Activate" (top right corner) to activate the key.

IX. Enable *Maps JavaScript API* and *Geocoding API* as follows:

   a. Navigation menu ➔ **APIs & Services ➔ Library ➔ Maps JavaScript API ➔ Enable**

   b. Navigation menu ➔ **APIs & Services ➔ Library ➔ Geocoding API ➔ Enable**

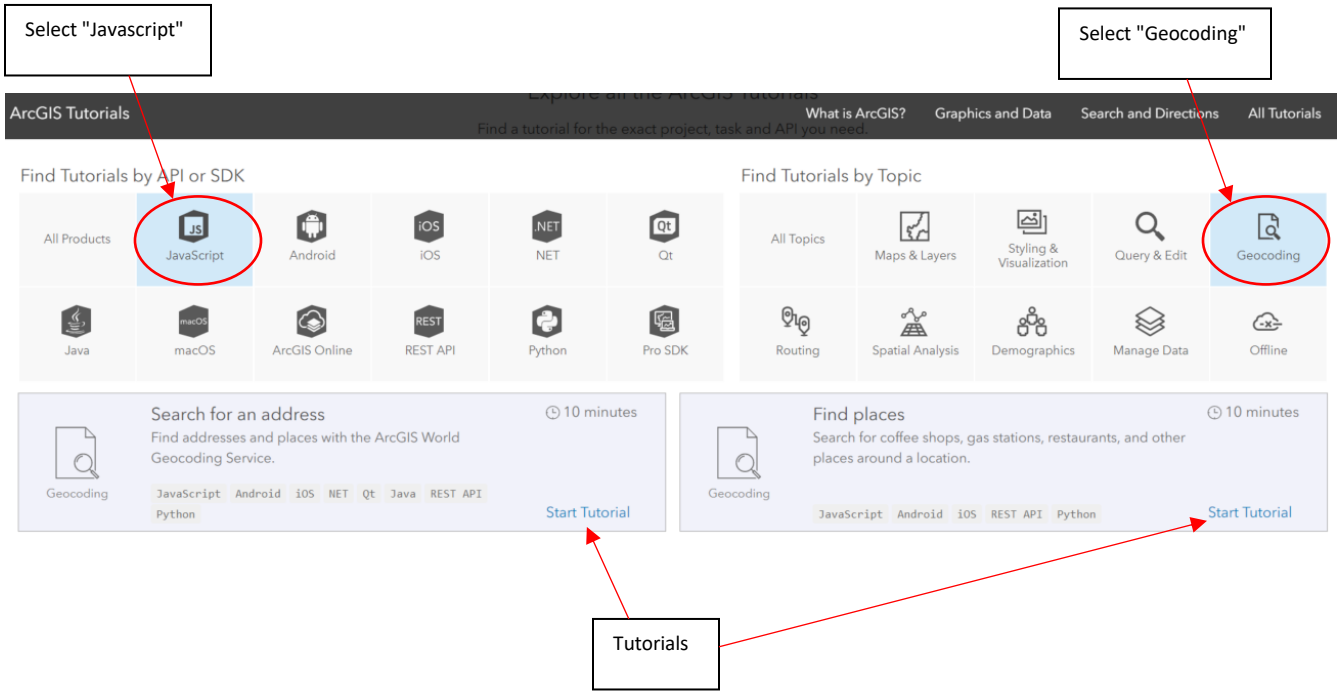
After acquiring a valid API Key, replace *APIKEY* with your valid API Key (encrypted string) in Google's API reference in <head> tag. Refresh the browser. Enter "Santa Fe New Mexico" in the *address* field and click on "Go" button. Correct coordinates for Santa Fe should be displayed and the map should be updated to show Santa Fe's map.

(6) ***Informational*** - ESRI (Environmental Systems Research Institute) seems to provide Open Source (or free) access to its APIs. Below is a link to tutorials etc. for developers:

https://developers.arcgis.com/labs/

There are a lot of tutorials to get you started on experimenting with API based webpages, especially, if you are interested in mapping-related applications.

For example, if you are interested in ***Java Script*** applications for ***Geocoding***, select these two topics to access/pinpoint the related material. You should see two related tutorials.



You can build applications on your own by following the examples given in the tutorials.