What You Will Learn

- How to process JSON in JavaScript
- How to asynchronously consume a web service using jQuery

Preparing Directories

- Create a folder in your personal drive for this activity (call it tutorial4).
- From Canvas → Tutorials → Tutorial 4, download the file tut4.zip and extract the files in the folder activity 5) you created in the first step.
- This activity walks you through the reading of JSON files, as well as the creation and consumption of JSON web services, which are simply JSON files delivered via HTTP.

1. Reading a JSON Data JavaScript

1 Open and examine tut4-1.html. Notice that it contains an empty <div> with the id of list. This we will populate in JavaScript.

2 Edit tut4-1.js as follows. Notice that is has a small JSON array already defined within the code.

```
window.addEventListener("load", function () {
    var text = '[{"id":100654,"name":"Alabama A & M", … }]';

    // turn JSON string into an actual JS array of objects
    var universities = JSON.parse(text);
    // display the data in the array
    var list = document.querySelector('#list');
    for (let i=0; i<universities.length; i++) {
        list.innerHTML += universities[i].name + '<br>';
    }
});
```

2. Consuming a JSON Web Service in JavaScript

1 Examine tut4-2.html.

You will be adding JavaScript in a separate file to consume the web service.

2 Test the web service by entering the following into a web browser:

```
http://www.randyconnolly.com/funwebdev/services/travel/countries.php
```

This returns a small subset of sample countries. Each country has a unique identifier (the iso property)

3 Add the following code to tut4-2.js and then test.

```
$(function () {

    // initialize countries select list

    displayCountries();
```

```
        function displayCountries() {
             // display animated loading GIF while data is being fetched

            $('.animLoading').show();

            var url = "http://www.randyconnolly.com/funwebdev/services/travel/countries.php";
             // now make asynchronous request for data from the web service

            $.get(url)
                .done(function (data) {
                      // loop through returned countries

                    for (let i=0; i<data.length; i++) {
                          // create option element and add to select list

                        var country = data[i];
                        var option = $('<option>',
                           {value: country.iso, text: country.name});
                        $("#countries").append(option);
                    }
                })
                .fail(function (jqXHR) {
                    alert("Error: " + jqXHR.status);
                })
                .always(function () {
                      // all done so now hide the animated loading GIF

                    $('.animLoading').fadeOut("slow");
                });
        }
    });
```

This should populate the first select list with a small list of countries. Now we will add an extra step: when use selects a country, we will make another request

4 Test the web service by entering the following into a web browser:

```
http://www.randyconnolly.com/funwebdev/services/travel/cities.php?iso=us
```

This returns a list of cities for the specified country (in this case, it is United States). Notice how each city also has a latitude and longitude, which we will later use for mapping purposes.

5 Add the following code to tut4-2.js and then test.

```
$(function () {
    // display countries select list
    displayCountries();

    // set up event handler for this select list
    $("#countries").on("change", displayCities);

    // responsible for retrieving a list of cities for a specific
    // country and then creating and populating a new select list
    // with these cities
```

```
function displayCities() {
    $('.animLoading').show();

    var url = "http://www.randyconnolly.com/funwebdev/services/travel/cities.php";
    var param = "iso=" + $('#countries').val();

     // only make web service request if the use has selected
     // an actual country
    if ($('#countries').val() != 0) {
        $.get(url, param)
            .done(function (data) {
                var select = $("<select id='cities'></select>");
                select.append("<option value=0>Select a city</option>");
                  // loop through an array using jquery's $.each() method
                $.each(data, function(index,city) {
                    select.append('<option value="' + city.id + '">'
                        + city.name + '</option>');
                });
                $("#results").empty().append(select);
            })
            .fail(function (jqXHR) {
                alert("Error: " + jqXHR.status);
            })
            .always(function () {
                // all done so now hide the animated loading GIF
                $('.animLoading').fadeOut("slow");
            });
    }
}
...
```

3. Web Services: Displaying a Google Map

1 Examine tut4-3.html.

You will be adding JavaScript in a separate file to consume the web service.

2 You will need a Google Maps JavaScript API key to do this next exercise. If you do not already have one, visit the following URL.

https://developers.google.com/maps/documentation/javascript/get-api-key

Once you create a key, ensure that you activate the key using the following URL.

https://developers.google.com/maps/gmp-get-started#enable-api-sdk

3 Modify the script tag in the head to use your Google Maps JavaScript API key.

```
<script type='text/javascript'

    src='https://maps.googleapis.com/maps/api/js?key=your key here'>
```

4 Add the following code to tut4-3.js and then test.

```
$(function () {
    $('.animLoading').show();

    var url = "http://www.randyconnolly.com/funwebdev/services/travel/cities.php";
    var param = "iso=CA";

    // make request for list of cities for specified country
    $.get(url, param)
        .done(function (data) {
             // loop through returned array of cities
            $.each(data, function(index,city) {
                 // create new empty list item
                var item = $('<li>');

                 // add lat and long info from web service to each
                 // list item using HTML5 data- attributes
                item.attr( "data-lat", city.latitude);
                item.attr( "data-long", city.longitude);
                item.html('<a href="#">' + city.name + '</a>');

                 // add list item to UL
                $("#cities").append(item);
            });
        })
        .fail(function (jqXHR) {
            alert("Error: " + jqXHR.status);
        })
        .always(function () {
             // all done so now hide the animated loading GIF
            $('.animLoading').fadeOut("slow");
        });
});
```

This should display a list of cities from Canada. If the list works, then the next step will display a map of the city when it is clicked.

5 Add the following code to the done() function in tut4-3.js.

```
$(function () {
   ...

                 // add lat and long info from web service to each
                 // list item using HTML5 data- attributes
                 item.attr( "data-lat", city.latitude);
                 item.attr( "data-long", city.longitude);
                 item.html('<a href="#">' + city.name + '</a>');

                 // add list item to UL
                 $("#cities").append(item);
             });

             // add handler for clicking on list items
            $("#cities li").on("click",  function () {
                displayMap($(this));
            });
```

6 Add the following nested function to tut4-3.js and test.

```
// display map for selected city
function displayMap(selectedCity) {
    // the lat and long of city is contained within
    // the clicked <li> element
    var ourLatLong = {lat: Number(selectedCity.attr("data-lat")) ,
                      lng: Number(selectedCity.attr("data-long"))};

    var ourMap = new google.maps.Map(document.getElementById('map'), {
        center: ourLatLong,
        scrollwheel: false,
        zoom: 13
    });
}
```

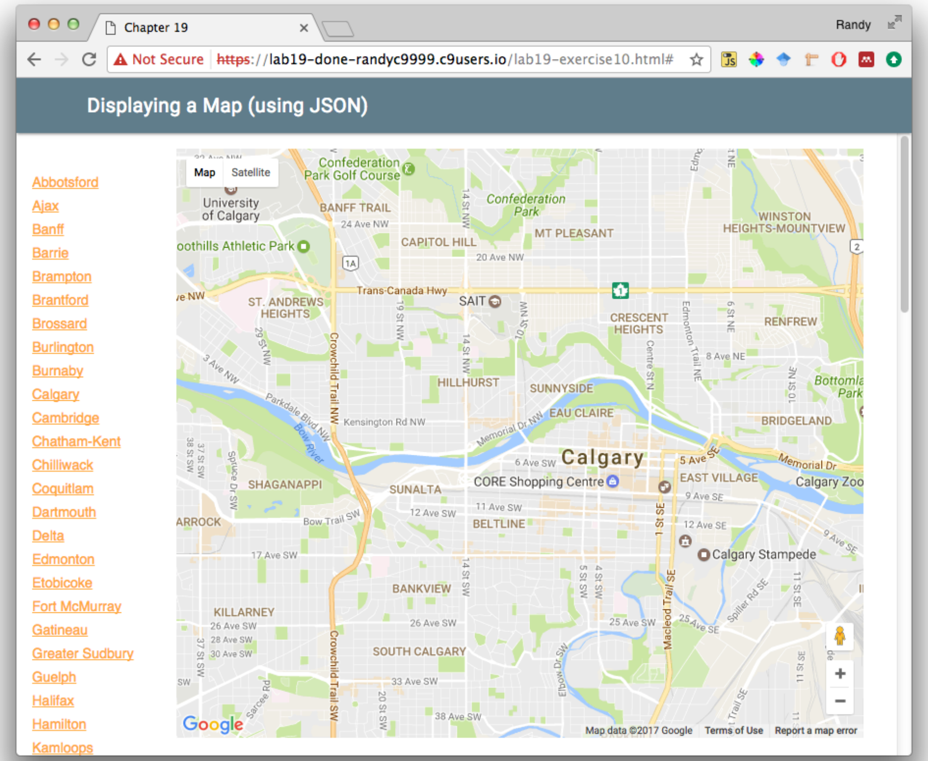

Figure 1 – Sample Output for Finished Exercise Tutorial 4 – Exercise 3.