# Working with the Geolocation API

Geolocation is about sharing the information about your location in the world – this can be done in a number of ways from using the IP address, the wireless connection, or from GPS. The latitude and longitude are available to JavaScript, which in turn processed on your web server to find local services or show your location on a map. Sharing with Geolocation is always opt-in – The Geolocation API states, "*User Agents must not send location information to Web sites without the express permission of the user.*" In this tutorial we will access this API and then use this data using Google Maps.

## Step 1: Developing with the `Geolocation API` – getting the latitude and longitude coordinates:

1. The geolocation API is published through a `geolocation` child object within the `navigator` object. If the object exists, geolocation services are available. You can test for the presence of geolocation using `Modernizr` or some JavaScript:

```
if (navigator.geolocation) {
    alert("Geolocation is available.");
} else {
  alert("I'm sorry, but geolocation services are not supported by your browser.");
}
```

2. Once you have tested whether or not your browser supports the geolocation you are ready to go. We are going to use jQuery for this tutorial. You can include this using the following:

   ```
   <script src="http://code.jquery.com/jquery-1.7.1.js"></script>
   ```

   or go to http://jquery.com/ and download a minified version. We will be taking an in depth look at jQuery in the following weeks, so today you will be given an example of all of the jQuery script that you need. You can use the `geo_starter.html`

3. Use the geolocation API to get the user's location when they click a button and output the values to the web page:

   ```
   <input type="button" id="go" value="Click Me To View Your Location" />
   ```

4. Add the following - when we load the document, we will append a ***click*** function to this button using jQuery.

```
$(document).ready(function () {
    // wire up button click
    $('#go').click(function () {
        // test geolocation
        if (navigator && navigator.geolocation) {
            navigator.geolocation.getCurrentPosition(geo_success, geo_error);
        } else {
            error('Geolocation is not supported.');
        }
    });
});
```

In the click function above, we obtain the **user's current location**, using the `getCurrentPosition()` method. This initiates an asynchronous request to detect the user's position, and queries the positioning hardware to get up-to-date information. When the position is determined, a specified callback routine is executed. A second callback is provided that will be executed if an error occurs.

5. The PositionError object returned contains the following codes as well as debugging error messages:
    a. PERMISSION_DENIED = 1
    b. POSITION_UNAVAILABLE = 2
    c. TIMEOUT = 3

    Include the following to handle the error messages:

```
function geo_error(err) {
    if (err.code == 1) {
        error('The user denied the request for location information.')
    } else if (err.code == 2) {
        error('Your location information is unavailable.')
    } else if (err.code == 3) {
        error('The request to get your location timed out.')
    } else {
        error('An unknown error occurred while requesting your location.')
    }
}

function error(msg) {
    alert(msg);
}
```

6. If we are successful at gathering the location information then we want to output them to the .html document. The following gets the coordinates and updates the inner html of `<p id = "lat">` and `<p id = "long">`

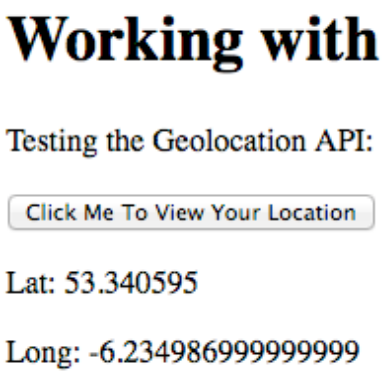
```
function geo_success(position) {
    printLatLong(position.coords.latitude, position.coords.longitude);
}

// output latitude and longitude
function printLatLong(lat, lng) {
    alert(lat + "    " + lng);
    $('#lat').html('<p>Lat: ' + lat + '</p>');
    $('#long').html('<p>Long: ' + lng + '</p>');
}
```

7. Test your output. The success callback is passed a position object that contains a coordinates object and a timestamp. The coordinates object contains the following:

    a. ***latitude***, decimal degrees
    b. ***longitude***, decimal degrees
    c. ***altitude***, meters above the ellipsoid
    d. ***accuracy***, meters
    e. ***altitudeAccuracy***, meters
    f. **heading**, direction of travel specified in degrees
    g. ***speed***, meters per second

    Of those seven, only latitude, longitude, and accuracy are guaranteed.



**Step 2: Reverse Geocoding - Converting the latitude and longitude into a human-friendly address:**

1. Here, we are going to use the `Google Maps JavaScript API` to determine a user's addressed based on the latitude and longitude. Turning geographic data like a street address and zip code into geographic coordinates such as latitude and longitude is called ***geocoding***. We are going to turn coordinates into an address - called ***reverse geocoding***.

2. Create a new .html document – you can use the `geo_reverseGecoding_starter.html`. Begin by adding the following scripts to your web page:

```
<script src="http://code.jquery.com/jquery-
     1.7.1.js"></script>

<script
src="http://maps.google.com/maps/api/js?sensor=false
"></script>
```

***3.*** Trigger getting the coordinates and address when the user clicks a button:

```
<input type="button" id="go" value="Click Me To View
Your Current Address" />
```

4. First we need to get our coordinates – the latitude and longitude from the previous sample. Adding a `click event to #go`, the `geo_success()`, `geo_error()`,and `error()` functions can be reused:

```
<script>
$(document).ready(function () {
    $('#go').click(function () {
        // test for presence of geolocation
        if (navigator && navigator.geolocation) {
            // make the request for the user's position
            navigator.geolocation.getCurrentPosition(geo_success, geo_error);
        } else {
           error('Geolocation is not supported.');
        }
    });
});

function geo_success(position) {
    printAddress(position.coords.latitude, position.coords.longitude);
}

function geo_error(err) {
    if (err.code == 1) {
        error('The user denied the request for location information.')
    } else if (err.code == 2) {
        error('Your location information is unavailable.')
    } else if (err.code == 3) {
        error('The request to get your location timed out.')
    } else {
        error('An unknown error occurred while requesting your location.')
    }
}
```

5. We are going to write some new functionality using the Google Maps API to reverse geocode our location and output it to `<p id = "local">`

```
// use Google Maps API to reverse geocode your location
function printAddress(latitude, longitude) {
    // Create a Google Maps Geocoder object
    var geocoder = new google.maps.Geocoder();

    // Change coordinates into a location
    var yourLocation = new google.maps.LatLng(latitude, longitude);

    // Use Google maps to find out about the location
    geocoder.geocode({ 'latLng': yourLocation }, function (results, status) {
        if (status == google.maps.GeocoderStatus.OK) {
            if (results[0]) {
                $('#local').html('Your Address:<br />' +
                    results[0].formatted_address);
            } else {
                error('Google did not return any results.');
            }
        } else {
            error("Reverse Geocoding failed due to: " + status);
        }
    });
}
```

The `google.maps.LatLng()` method creates a new `Google LatLng` object that is passed into geocode() in order to get the address. The geocode() method is asynchronous and contains two parameters, one for the ***results*** and the other for the ***status code***. If the status is OK, then it's safe to parse the array of GeocoderResults objects stored in the results variable. Next, check for a GeocoderResults object and, if it exists, append the `formatted_address` property to the inner html of the `<p id = "local">`.

**Step 3: Get your location and display it on a map of your immediate surroundings**

1. This example will have similar elements to the pervious samples. We also have to add a div to hold the Google Map `<div id="map">`

# Working with Geolocation

Testing the Geolocation API:

Click Me To View Your Current Address

Your Address:
Mayor Street Lower, I.F.S.C., Dublin 1, Ireland


This page was created by for ARIA for testing the Geolocation API with Google maps

2. Create a global `map variable` and update your setup so that it creates a new map that is pointed at Ireland. The following map types are available in the Google Maps API:
    a. MapTypeId.ROADMAP displays the default road map view
    b. MapTypeId.SATELLITE displays Google Earth satellite images
    c. MapTypeId.HYBRID displays a mixture of normal and satellite views
    d. MapTypeId.TERRAIN displays a physical map based on terrain information.

Map options are described in depth at http://code.google.com/apis/maps/documentation/javascript/maptypes.html

```
// Google Maps global map variable
var map;

$(document).ready(function () {
    var myOptions = {
      center: new google.maps.LatLng(53.5, -7),
      zoom: 4,
      mapTypeId: google.maps.MapTypeId.ROADMAP
    };

    map = new google.maps.Map(document.getElementById("map"), myOptions);


    $('#go').click(function () {
        // test for presence of geolocation
        if (navigator && navigator.geolocation) {
            // make the request for the user's position
            navigator.geolocation.getCurrentPosition(geo_success, geo_error);
        } else {
           error('Geolocation is not supported.');
        }
    });
});
```

3. The next step is to change the map location and zoom once the user clicks the button and provides their coordinates. You need to edit your `printAddress()` function to achieve this.

4. Finally, add a marker to your location.

```
// To add the marker to the map set it up then call setMap();
var marker = new google.maps.Marker({
    position: yourLocation,
    title:"Hello World!"
});

marker.setMap(map);
```

5. Test your output:

# Working with Geolocation

Testing the Geolocation API:

Click Me To View Your Current Address

Your Address:
Mayor Street Lower, I.F.S.C., Dublin 1, Ireland


This page was created by for ARIA for testing the Geolocation API with Google maps

6. Push your solution to your ARIA-Git-Project ***GitHub*** project fork.

***Further References:***

W3C Geolocation API:
http://www.w3.org/TR/geolocation-API/

Google Maps JavaScript API:
http://code.google.com/apis/maps/documentation/javascript/

Mozilla Developer Network – Using Geolocation:
https://developer.mozilla.org/en/Using_geolocation

Mozilla Developer Network – Geolocation Demo Studio
https://developer.mozilla.org/en-US/demos/tag/tech%3Ageolocation/

You are Here (And so is Everybody Else) – Dive into HTML5:
http://diveintohtml5.info/geolocation.html

Google Location Based Mobile Websites: http://code.google.com/p/geo-location-javascript/

HTML5 Mobile Web Apps: Positioning with Geolocation:
http://mobile.tutsplus.com/tutorials/mobile-web-apps/html5-geolocation/

HTML5 Cookbook – Chapter 8 – Getting Started with HTML5 Geolocation
http://www.netmagazine.com/tutorials/getting-started-html5-geolocation

Polymaps: http://polymaps.org/

SimpleGeo: https://simplegeo.com/developers/