Using Geolocation in HTML

Have you ever need to display your user's location on a map in your web application? HTML 5 adds a geolocation object to help make locating the current user's latitude and longitude quick and easy. Once you have this information, you can use a map API such as Google Maps or Microsoft's Bing Maps to display that latitude and longitude on a graphical map. This blog post explores how to use this new object to get a user's current position.

Get Latitude and Longitude

The first thing you should do is check to ensure the user's browser supports the geolocation object. You can use the following code to check if it is not null.

```
if (navigator.geolocation) {
  // Do Something Here
}
```

Once you determine the geolocation object is available, call the getCurrentPosition() method and pass in a callback function to which the getCurrentPosition() method passes a **Coordinates** object.

```
navigator.geolocation.getCurrentPosition(currentLocation);
```

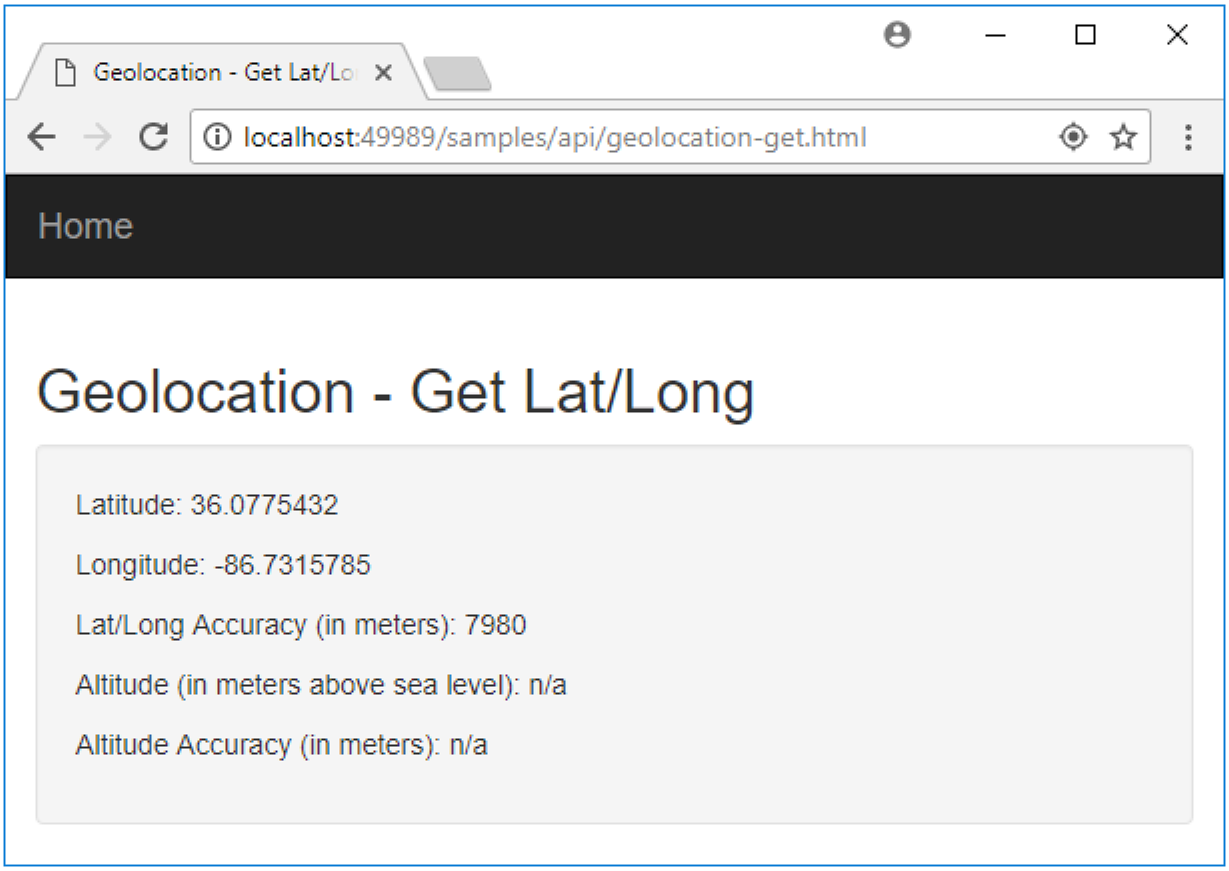
The currentLocation() function you pass to the getCurrentPosition() method receives a **Coordinates** object from which you can extract the latitude, longitude, and other properties. The code snippet below shows some HTML, and the currentLocation() function that inserts the latitude and longitude into that HTML using jQuery.

```
<p>Latitude: <span id="lat"></span></p>
<p>Longitude: <span id="long"></span></p>

function currentLocation(coordinates) {
  $("#lat").text(coordinates.coords.latitude);
  $("#long").text(coordinates.coords.longitude);
}
```

Display Coordinate Properties

To give you a good sense of how this works, let's create a simple web page that looks like **Figure 1**. This web page uses the Bootstrap CSS framework. However, feel free to use whatever CSS you like.



Geolocation - Get Lat/Long

Latitude: 36.0775432

Longitude: -86.7315785

Lat/Long Accuracy (in meters): 7980

Altitude (in meters above sea level): n/a

Altitude Accuracy (in meters): n/a

Figure 1: Display Latitude, Longitude and Accuracy

The HTML code for this page is presented below. A simple navigation bar is added at the top to navigate back to an index page. Several <p> tags are used with <span> elements inside of them to display the latitude, longitude, accuracy, altitude and the altitude accuracy. Instructions to download this complete sample are at the end of this blog post.

```
<!doctype html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Geolocation - Get Lat/Long</title>

  <link href="content/bootstrap.min.css" rel="stylesheet" />
</head>
<body>
  <nav class="navbar navbar-fixed navbar-inverse" role="navigation">
    <div class="container">
      <div class="navbar-header">
        <a href="index.html" title="Home" class="navbar-brand">
          Home
        </a>
      </div>
    </div>
  </nav>
  <div class="container">
    <h2>Geolocation - Get Lat/Long</h2>
    <div class="row">
      <div class="col-xs-12">
        <div class="well">
          <p>Latitude:
            <span id="lat"></span>
          </p>
          <p>Longitude:
            <span id="long"></span>
          </p>
          <p>Lat/Long Accuracy (in meters):
            <span id="latlongaccuracy"></span>
          </p>
          <p>Altitude (in meters above sea level):
            <span id="altitude"></span>
          </p>
          <p>Altitude Accuracy (in meters):
            <span id="altitudeaccuracy"></span>
          </p>
        </div>
      </div>
    </div>
  </div>

  <script src="scripts/jquery-3.3.1.min.js"></script>
  <script>

  </script>
</body>
</html>
```

To call the getCurrentPosition() method and retrieve the **Coordinates** object, write the code within the empty <script> tag. First, add a closure for the callback function that the getCurrentPosition() will call.

```
var geoController = (function () {
  // Private Functions
  function currentLocation(coordinates) {
    $("#lat").text(coordinates.coords.latitude);
    $("#long").text(coordinates.coords.longitude);
    $("#latlongaccuracy").text(coordinates.coords.accuracy);
    $("#altitude").text(coordinates.coords.altitude ?
                        coordinates.coords.altitude : "n/a");
    $("#altitudeaccuracy").text(
             coordinates.coords.altitudeAccuracy ?
             coordinates.coords.altitudeAccuracy : "n/a");
  }

  // Public Functions
  return {
    currentLocation: function (coordinates) {
      currentLocation(coordinates);
    }
  }
})();
```

Next, add the code that calls this currentLocation() function when the document is ready.

```
$(document).ready(function () {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(
         geoController.currentLocation);
  }
});
```

Try it Out

You should now be able to run this web page. Your browser will prompt you that this web page is asking to know your location. There will be two options: to allow this page to access the geolocation coordinates or decline access. For now, go ahead and allow access to see your latitude, longitude, and other properties.

Add Error Handling

You do not want to assume that the geolocation object will work. There are a few different things that can go wrong: the user may choose not to allow access to the geolocation object, the call to getCurrentPosition() may timeout, the current user's location may not be able to be determined, or other types of errors may occur. In case an error does occur, make sure to set up an error message to display on the screen. Here’s how it’s done.

Add the following <div> just after the other <div> you added that contains the latitude, longitude, and other properties. It is within this <div> tag that you are going to display the error message. This <div> uses the Bootstrap "alert" style to make it stand out on the page.

```
<div class="row">
  <div class="col-xs-12">
    <div id="geoError" class="alert alert-danger hidden">
    </div>
  </div>
</div>
```

You are going to need a method in the geoController closure to display an error message within this error <div> tag. Add a method named showErrorMessage() to which you pass a message to display. It will insert this message within the <div> and remove the "hidden" class to display the error message.

```
function showErrorMessage(msg) {
  $("#geoError").text(msg);
  $("#geoError").removeClass("hidden");
}
```

Add another method within the geoController closure you created named handleLocationError(). This method is passed as a callback to the second parameter of the getCurrentPosition() method.

```
function handleLocationError(error) {
  var msg = "";

  msg = error.message;
  console.log(msg);
  switch (error.code) {
    case error.PERMISSION_DENIED:
      msg = "User does not want to display their location."
      break;
    case error.POSITION_UNAVAILABLE:
      msg = "Can't determine user's location."
      break;
    case error.TIMEOUT:
      msg = "The request for geolocation information timed out."
      break;
    case error.UNKNOWN_ERROR:
      msg = "An unknown error occurred."
      break;
  }

  showErrorMessage(msg);
}
```

Add these new methods to the return statement in the geoController closure.

```
return {
  currentLocation: function (coordinates) {
    currentLocation(coordinates);
  },
  handleLocationError: function (error) {
    handleLocationError(error);
  },
  showErrorMessage: function (msg) {
    showErrorMessage(msg);
  }
}
```

Finally, re-write the document ready function and pass the handleLocationError() method as the second parameter to the getCurrentPosition() method. Also, add an else statement to display a message if the geolocation object is not valid for this browser.

```
$(document).ready(function () {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(
      geoController.currentLocation,
      geoController.handleLocationError);
  }
  else {
    geoController.showErrorMessage(
        "Geolocation is not available for this browser.");
  }
});
```

Geolocation Options

There is a third parameter you may pass to the getCurrentPosition() method. This third parameter is an optional PositionOptions object. This object contains three properties, of which you may set one, two, or all three of them. One thing to note; these options are only applied if you are running in HTTPS mode in your browser. The properties and their meaning are listed in the table below.

enableHighAccuracy	If set to true, it informs the device to attempt to obtain a more accurate position. This can slow down the time it takes to retrieve the coordinates.
timeout	The maximum length of time to wait before throwing an exception. This value is expressed in milliseconds.

maximumAge	The maximum age of the last cached position that is acceptable to return. This value is expressed in milliseconds. Set to 0 to not allow the device to cache positions.

To use these options, add a new variable named *options*, and set it equal to a literal object that you can see in the code in bold below. Pass this object as the third parameter to the getCurrentPosition() method.

```
$(document).ready(function () {
  if (navigator.geolocation) {
    var options = {
      enableHighAccuracy: true,
      timeout: 10000,     // 10 seconds
      maximumAge: 300000  // No position older than 5 minutes
    };

    navigator.geolocation.getCurrentPosition(
      geoController.currentLocation,
      geoController.handleLocationError,
      options);
  }
  else {
    geoController.showErrorMessage(
      "Geolocation is not available for this browser.");
  }
});
```

Summary

In this blog post, you learned to retrieve a user's current latitude and longitude from the geolocation object available in HTML 5. You saw how to handle errors and pass some options to the getCurrentPosition() method. There are additional methods available on the geolocation object. These methods will help you perform real-time monitoring of a user's position.

Getting the Sample Code

You may download the sample code by navigating to www.pdsa.com/downloads. Select "PDSA/Fairway Blog" from the Category drop-down. Then select "Using Geolocation in HTML" from the Item drop-down.