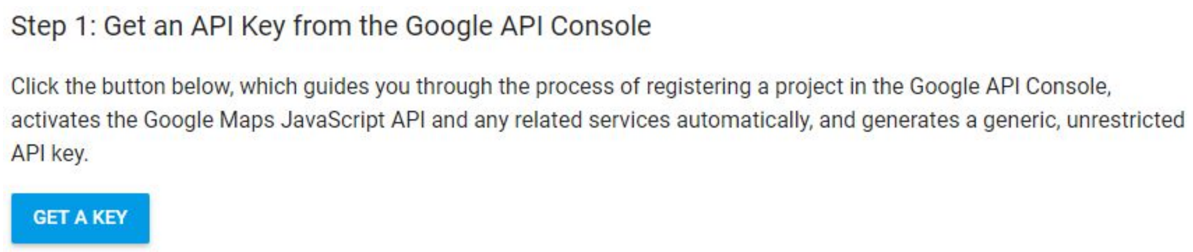
Google Maps Tutorial

General Setup

Overview: This section of the tutorial includes basic setup necessary to use Google Maps and potential methods to utilize the code on the web and on mobile devices.

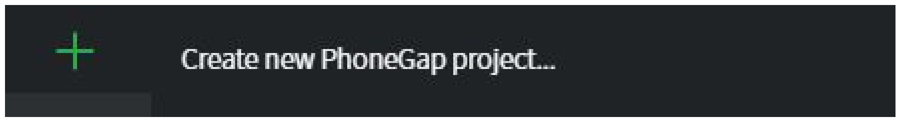
1. Get an Google Maps API key
    a. Go to https://developers.google.com/maps/documentation/javascript/get-api-key
    b. Click on the button that says "GET A KEY"



Step 1: Get an API Key from the Google API Console

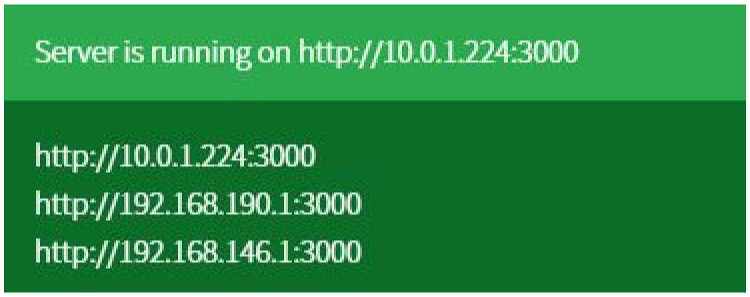
Click the button below, which guides you through the process of registering a project in the Google API Console, activates the Google Maps JavaScript API and any related services automatically, and generates a generic, unrestricted API key.

    c. Follow the prompts to create a new project and receive an API key
    d. Copy and paste the key into a blank document or note for later
2. Set up a PhoneGap Project (optional-convenient to run on mobile devices)
    a. Open PhoneGap Desktop
    b. Press the + button to create a new project



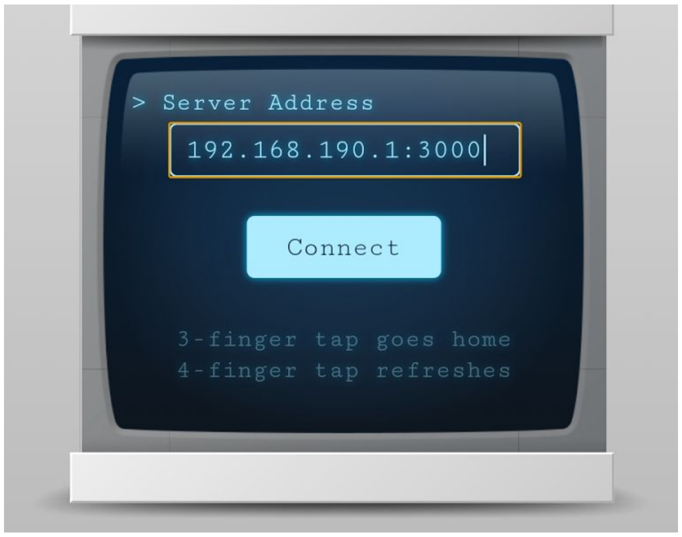
    c. Create a "Blank" project
    d. Choose the desired location path and name
    e. Modify the main HTML file
        i. Open Adobe Dreamweaver (or another text editor)
        ii. Navigate to the project location
        iii. Go to the www folder
        iv. Open "index"
    f. Test code

i. Press one of the IP addresses at the bottom of the PhoneGap Desktop application to run on your computer



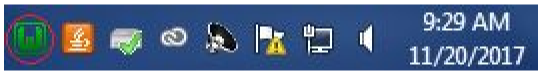
```
Server is running on http://10.0.1.224:3000

http://10.0.1.224:3000
http://192.168.190.1:3000
http://192.168.146.1:3000
```

ii. Input the IP address from PhoneGap Desktop into the text box labeled "Server Address" on the mobile application to test on a mobile device



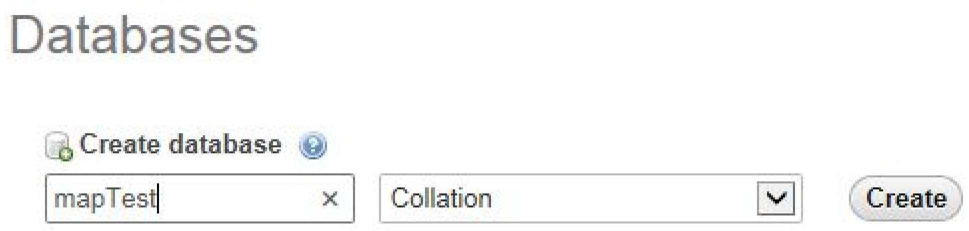
```
> Server Address
192.168.190.1:3000
```

3. Set up MySQL Wampserver for inputting/outputting data (use if you wish to input and output stored data)
   a. Open Wampserver64
   b. Press the small icon from the lower right hand corner and open phpMyAdmin



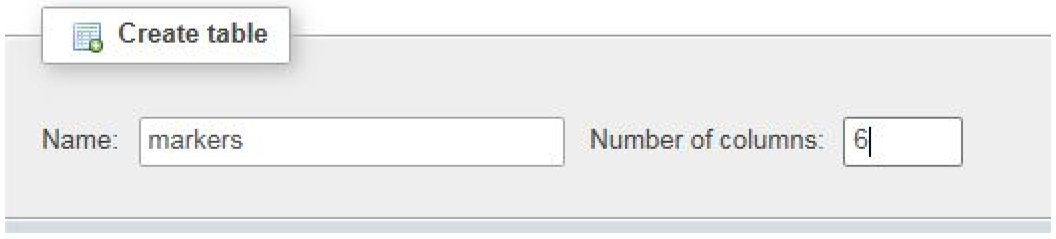
   c. Login with username root and no password

d. Go to New and create a Database with your desired name



mapTest

e. Go into your database and create a table named “markers” with attributes from the map: id, name, address, lat, lng, and type



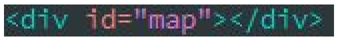
Name: markers

Number of columns: 6

Creating a Map With Markers

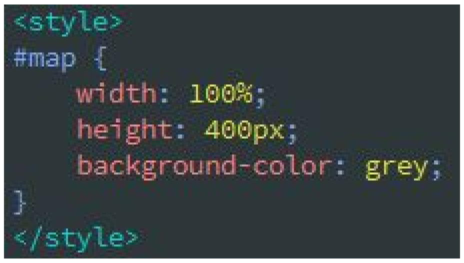
Overview: This section of the tutorial outlines how to create a basic map using HTML and place markers at specific longitudes and latitudes.

1. Either create a new PhoneGap project (see “General Setup”) or create a new HTML document in the text editor of your choice
2. Create an area for the map
   a. Make a div for the map



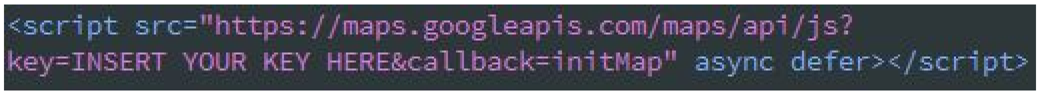
```
<div id="map"></div>
```

   b. Style the div as desired (the following code will make it full width, with a height of 400px, and a background color of gray)



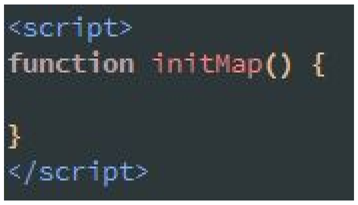
```
<style>
#map {
    width: 100%;
    height: 400px;
    background-color: grey;
}
</style>
```

   c. Import your map using your API key



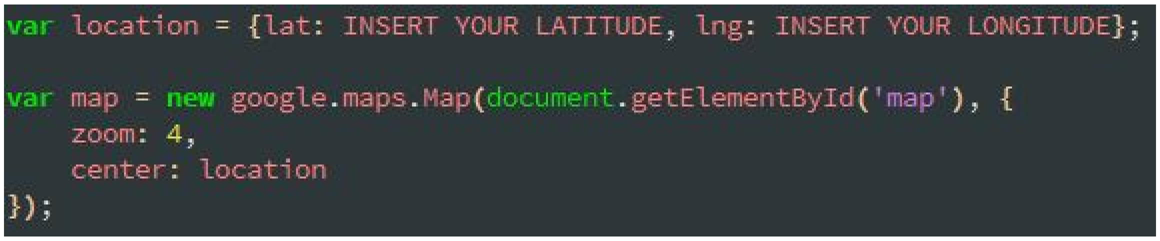
```
<script src="https://maps.googleapis.com/maps/api/js?
key=INSERT YOUR KEY HERE&callback=initMap" async defer></script>
```

3. Create the map
   a. Initiate the map with Javascript



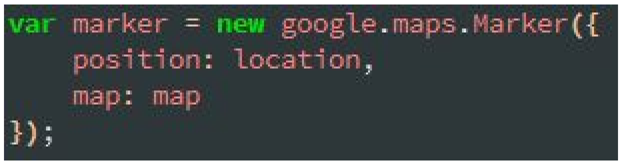
```
<script>
function initMap() {

}
</script>
```

   b. Within the initMap() function, construct a Google Maps object with your desired location (latitude and longitude)



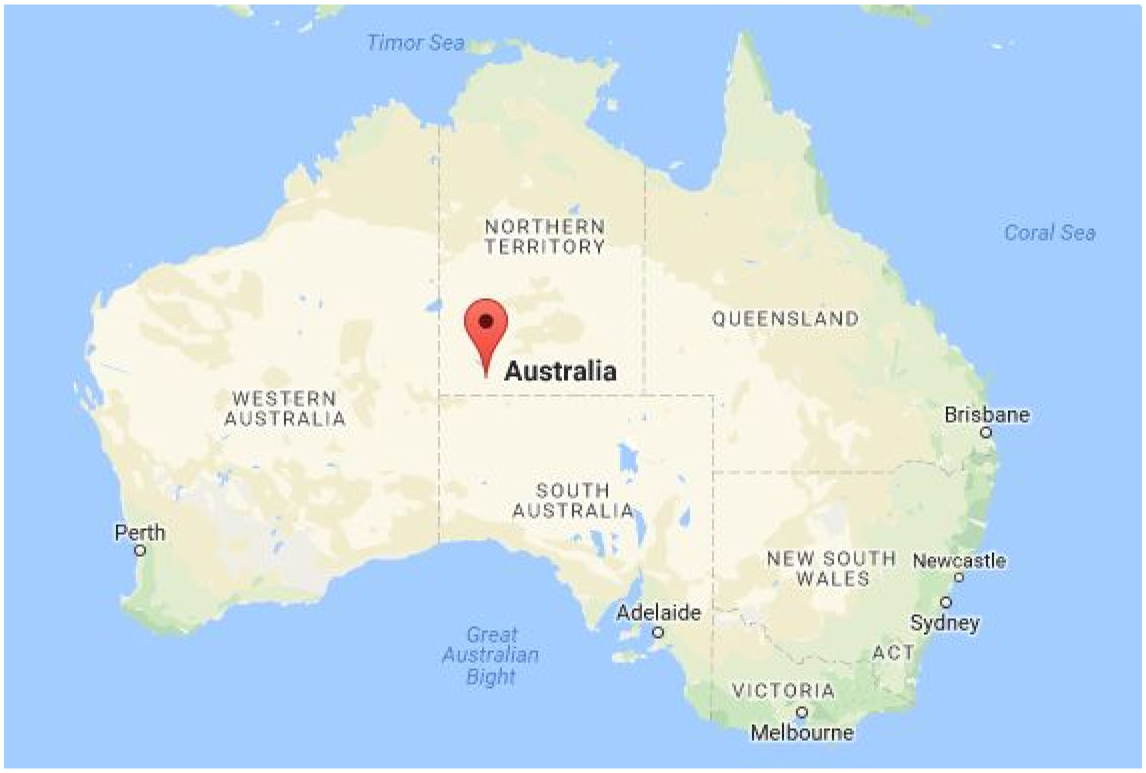
```
var location = {lat: INSERT YOUR LATITUDE, lng: INSERT YOUR LONGITUDE};

var map = new google.maps.Map(document.getElementById('map'), {
    zoom: 4,
    center: location
});
```

c. Also within the initMap() function, place a marker at your desired location



```
var marker = new google.maps.Marker({
    position: location,
    map: map
});
```

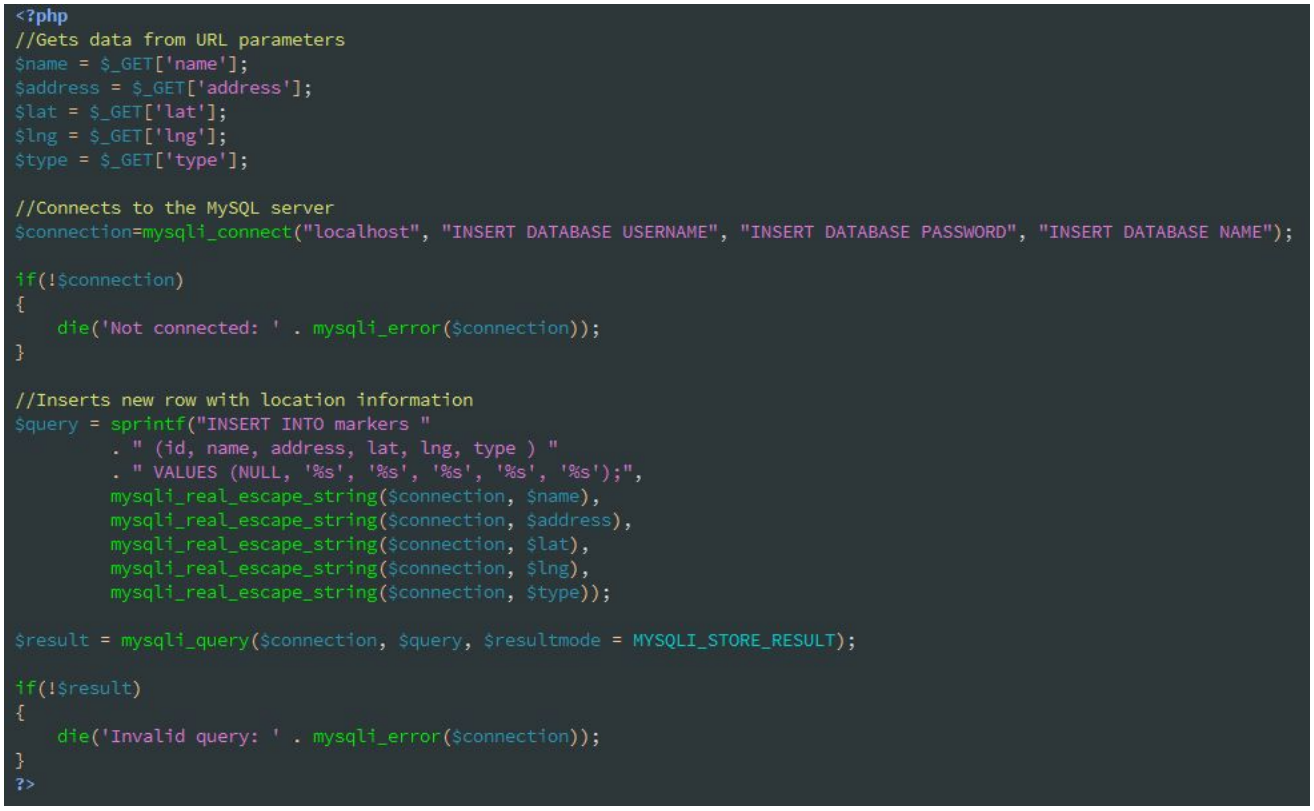
4. Run and test code
   a. Use PhoneGap or your desired text editor to preview the map in a browser/application viewer (see “General Setup” for PhoneGap instructions)
   b. Look to see the marker in your desired location



Inputting Data to a MySQL Database

Overview: This section of the tutorial explains how to set up a map on which you can place markers that are stored in a MySQL database

1. Either create a new PhoneGap project (see "General Setup") or create a new HTML document in the text editor of your choice
2. Setup a MySQL server with a table for location attributes (see "General Setup" for more information)
3. Create php file in C:\wamp64\www for connecting your HTML code to the MySQL database (copy code from below)



```
<?php
//Gets data from URL parameters
$name = $_GET['name'];
$address = $_GET['address'];
$lat = $_GET['lat'];
$lng = $_GET['lng'];
$type = $_GET['type'];

//Connects to the MySQL server
$connection=mysqli_connect("localhost", "INSERT DATABASE USERNAME", "INSERT DATABASE PASSWORD", "INSERT DATABASE NAME");

if(!$connection)
{
    die('Not connected: ' . mysqli_error($connection));
}

//Inserts new row with location information
$query = sprintf("INSERT INTO markers "
         . " (id, name, address, lat, lng, type ) "
         . " VALUES (NULL, '%s', '%s', '%s', '%s', '%s');",
         mysqli_real_escape_string($connection, $name),
         mysqli_real_escape_string($connection, $address),
         mysqli_real_escape_string($connection, $lat),
         mysqli_real_escape_string($connection, $lng),
         mysqli_real_escape_string($connection, $type));

$result = mysqli_query($connection, $query, $resultmode = MYSQLI_STORE_RESULT);

if(!$result)
{
    die('Invalid query: ' . mysqli_error($connection));
}
?>
```

4. In the HTML file, create a map as demonstrated above in the "Creating a Map with Markers" section
5. Create table and save message (outside of initMap() function and <script> tags)

a. Create a table with rows and columns for each variable in your mySQL table

```
<div id="form">
  <table>
  <tr><td>Name:</td> <td><input type="text" id="name"/> </td> </tr>
  <tr><td>Address:</td> <td><input type="text" id="address"/> </td> </tr>
  <tr><td>Type:</td> <td><select id="type"> +
             <option value="INSERTYOURTYPE1" SELECTED>INSERT TYPE 1</option>
             <option value="INSERTYOURTYPE2">INSERT TYPE 2</option>
             </select> </td></tr>
             <tr><td></td><td><input type="button" value="Save" onclick="saveData()"/></td></tr>
  </table>
</div>
```

b. Display text upon saving the inputted data

```
<div id="message">Location saved</div>
```

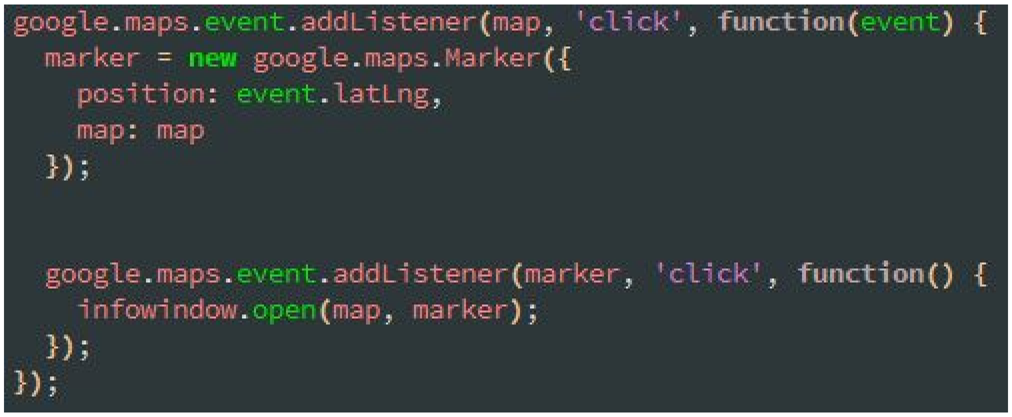
6. Define variables (outside of initMap() function, inside the <script> tags)

```
var map;
var marker;
var infowindow;
var messagewindow;
```

7. Within the initMap() function, create objects for the info window for the markers and saving the inputted information

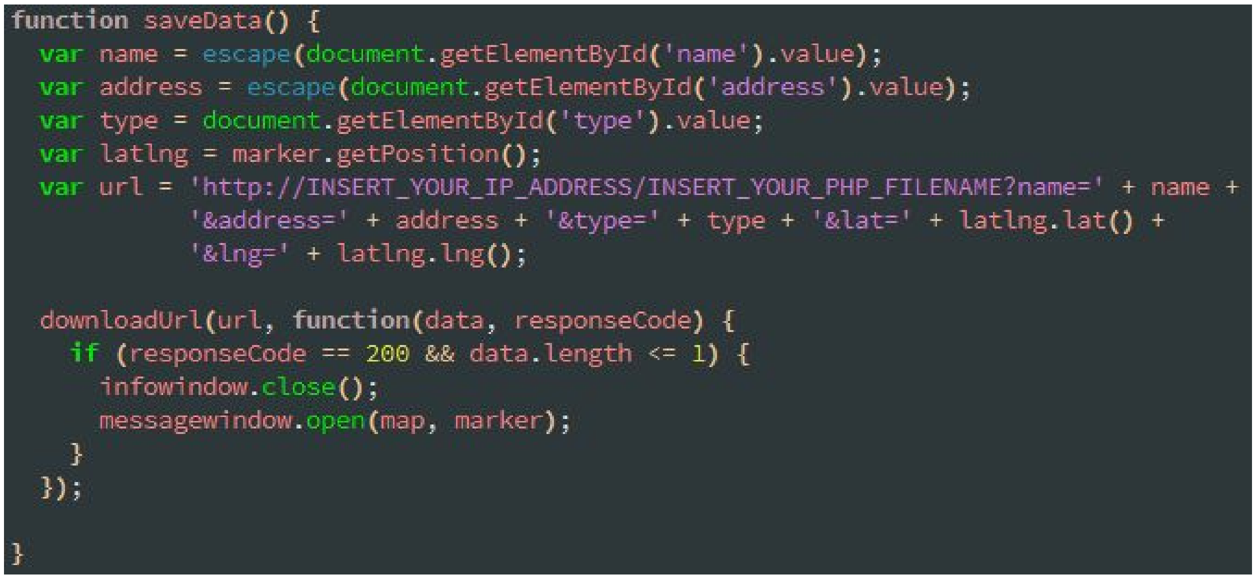
```
infowindow = new google.maps.InfoWindow({
  content: document.getElementById('form')
});

messagewindow = new google.maps.InfoWindow({
  content: document.getElementById('message')
});
```

8. Define click listeners so the markers can be selected



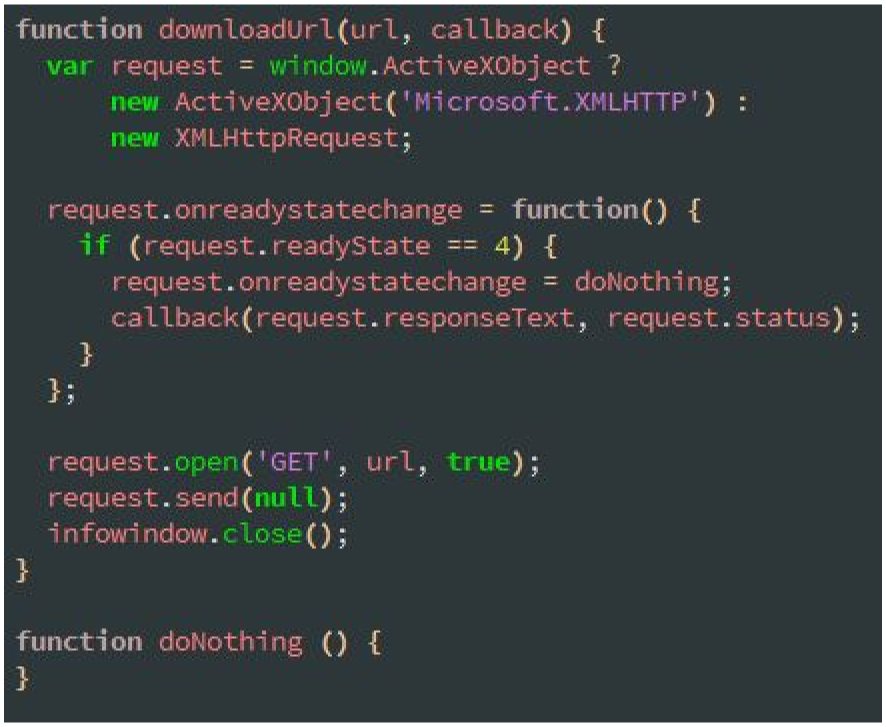
```
google.maps.event.addListener(map, 'click', function(event) {
  marker = new google.maps.Marker({
    position: event.latLng,
    map: map
  });


  google.maps.event.addListener(marker, 'click', function() {
    infowindow.open(map, marker);
  });
});
```

9. Within the <script> tags, outside the initMap() function, save data into MySQL database using an XMLHTTPRequest object (also checks that the file was retrieved based on the response code)



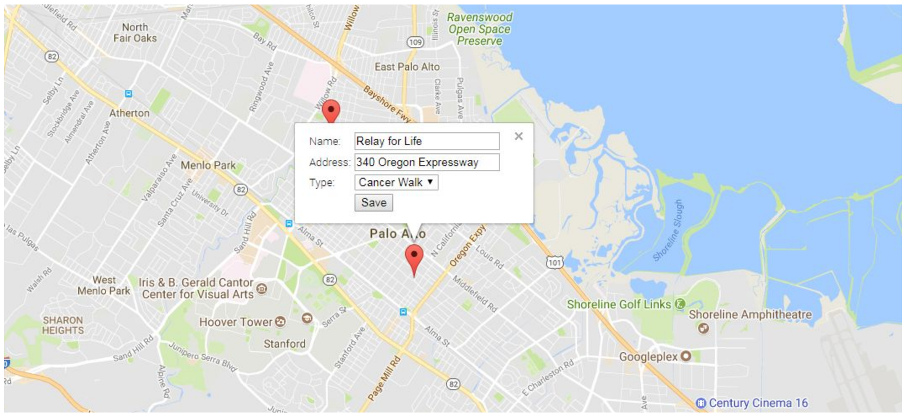
```
function saveData() {
  var name = escape(document.getElementById('name').value);
  var address = escape(document.getElementById('address').value);
  var type = document.getElementById('type').value;
  var latlng = marker.getPosition();
  var url = 'http://INSERT_YOUR_IP_ADDRESS/INSERT_YOUR_PHP_FILENAME?name=' + name +
            '&address=' + address + '&type=' + type + '&lat=' + latlng.lat() +
            '&lng=' + latlng.lng();

  downloadUrl(url, function(data, responseCode) {
    if (responseCode == 200 && data.length <= 1) {
      infowindow.close();
      messagewindow.open(map, marker);
    }
  });

}
```

10. Define downloadUrl and doNothing functions to complete XMLHTTP file retrieval and saving



```
function downloadUrl(url, callback) {
  var request = window.ActiveXObject ?
      new ActiveXObject('Microsoft.XMLHTTP') :
      new XMLHttpRequest;

  request.onreadystatechange = function() {
    if (request.readyState == 4) {
      request.onreadystatechange = doNothing;
      callback(request.responseText, request.status);
    }
  };

  request.open('GET', url, true);
  request.send(null);
  infowindow.close();
}

function doNothing () {
}
```

11. Run and test code
    a. Use PhoneGap or your desired text editor to preview the map in a browser/application viewer (see "General Setup" for PhoneGap instructions)
    b. Press on a desired location and input your data for the marker in your pop-up window

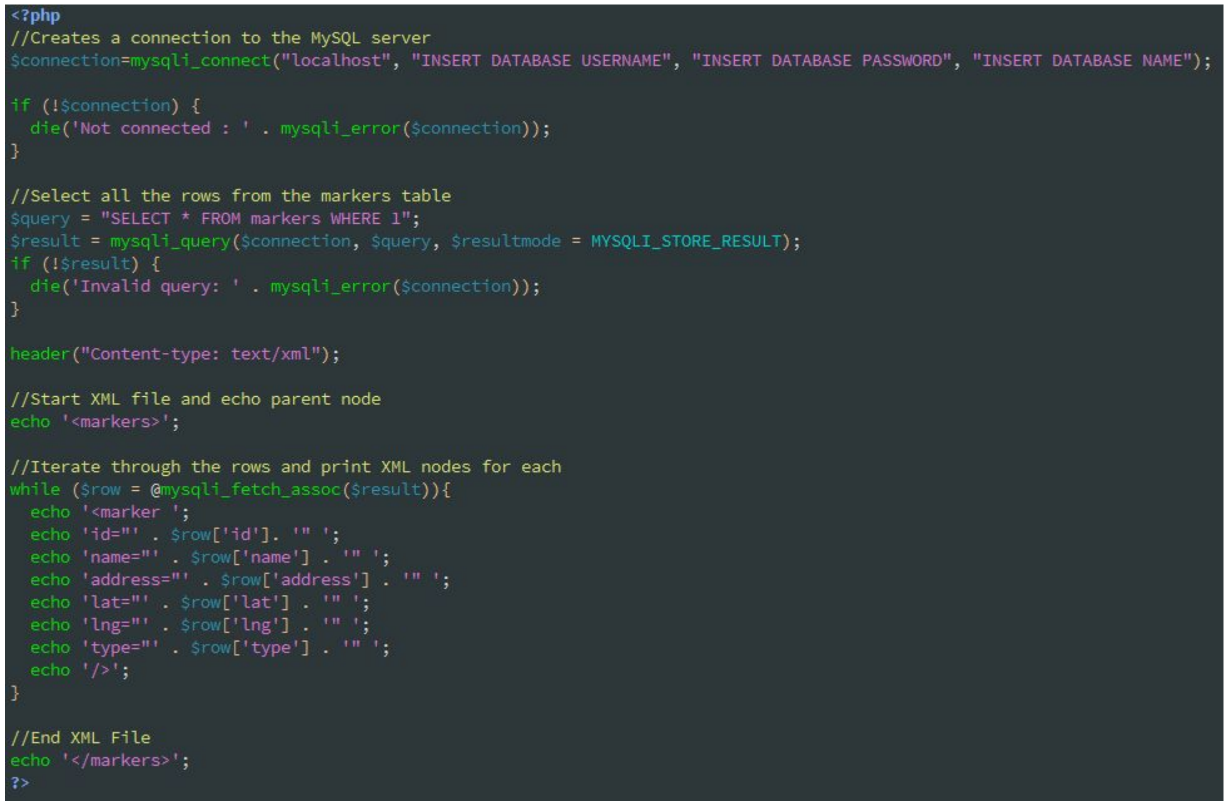


    c. Check your mySQL database and see if the values were inputted into the "markers" table

Outputting Data from a MySQL Database

Overview: This section of the tutorial explains how to output data from a MySQL table as markers on a map.

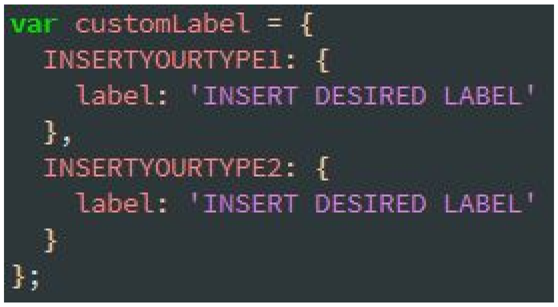
1. Either create a new PhoneGap project (see “General Setup”) or create a new HTML document in the text editor of your choice
2. Set up a MySQL server with a table for location attributes (see “General Setup” for more information)
    a. Note: if you have already completed the “Inputting Data to a MySQL Database” section, you can just output the data values from that server
    b. If you have not already used the previous section of the tutorial to create a MySQL data table with entries, input new locations into your MySQL table
3. Create php file in C:\wamp64\www for outputting MySQL data (copy code from below)
    a. Use PHP’s echo function to output XML



```
<?php
//Creates a connection to the MySQL server
$connection=mysqli_connect("localhost", "INSERT DATABASE USERNAME", "INSERT DATABASE PASSWORD", "INSERT DATABASE NAME");

if (!$connection) {
  die('Not connected : ' . mysqli_error($connection));
}

//Select all the rows from the markers table
$query = "SELECT * FROM markers WHERE 1";
$result = mysqli_query($connection, $query, $resultmode = MYSQLI_STORE_RESULT);
if (!$result) {
  die('Invalid query: ' . mysqli_error($connection));
}

header("Content-type: text/xml");

//Start XML file and echo parent node
echo '<markers>';

//Iterate through the rows and print XML nodes for each
while ($row = @mysqli_fetch_assoc($result)){
  echo '<marker ';
  echo 'id="' . $row['id']. '" ';
  echo 'name="' . $row['name'] . '" ';
  echo 'address="' . $row['address'] . '" ';
  echo 'lat="' . $row['lat'] . '" ';
  echo 'lng="' . $row['lng'] . '" ';
  echo 'type="' . $row['type'] . '" ';
  echo '/>';
}

//End XML File
echo '</markers>';
?>
```

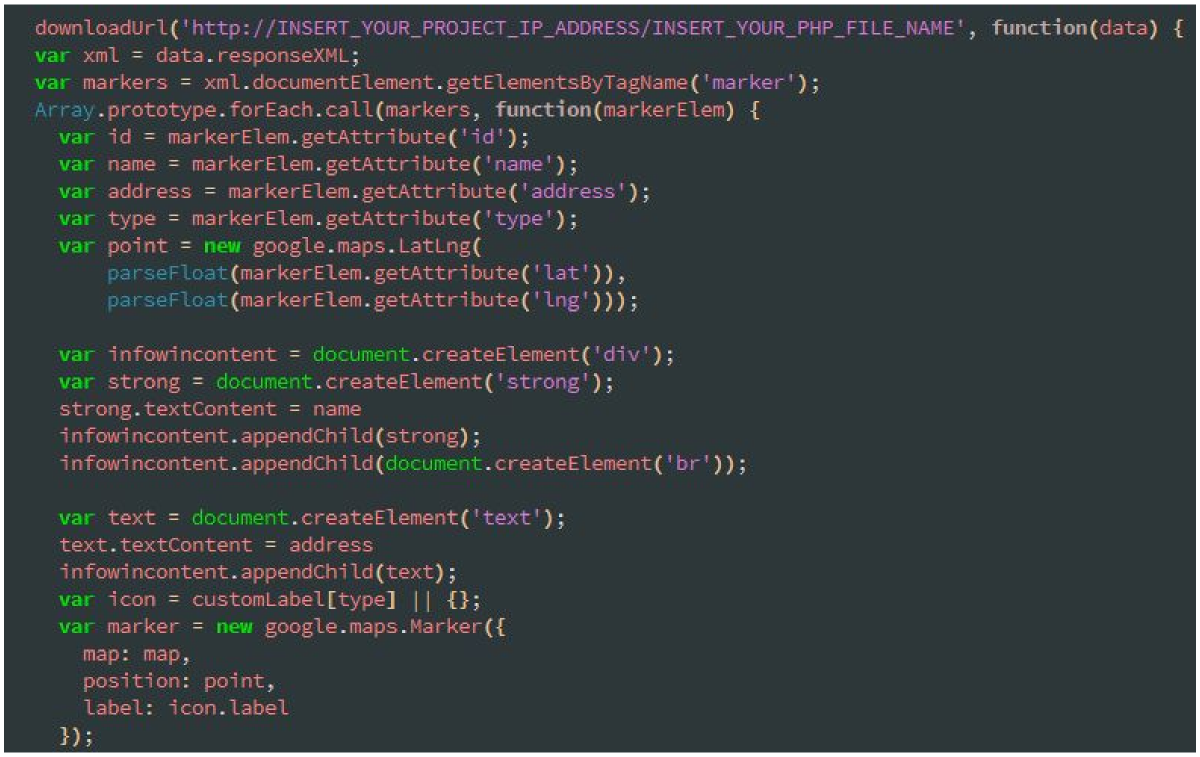
This code will connect to the database and execute a SELECT query on the markers table. Then, it echoes the parent markers node and iterates through the query results. Finally, it echoes the XML node for each row

and sends the data through the parseToXML function to output the final XML document.
    b. Test php script by calling it from the browser and checking if it outputs the data for each marker
4. In the HTML file, create a basic map framework (see “Creating a Map with Markers” section)
5. Within the <script> tags but outside of the initMap() function, create your custom markers with labels



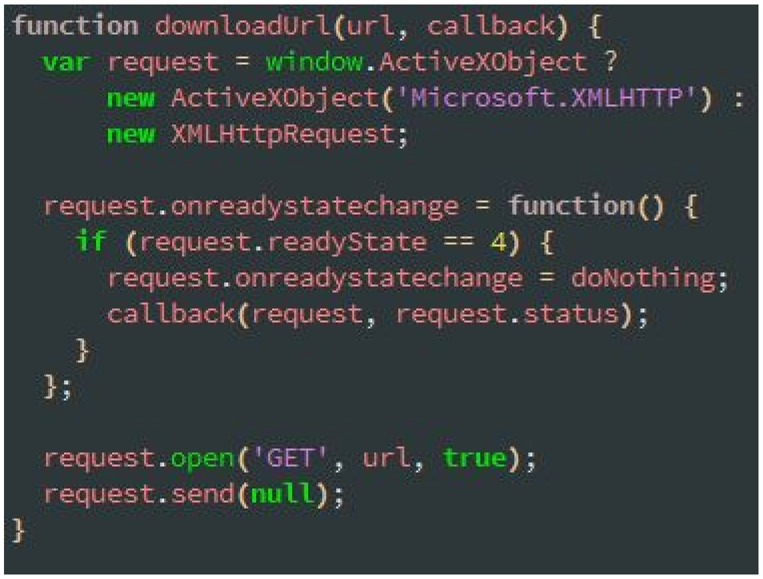
```
var customLabel = {
  INSERTYOURTYPE1: {
    label: 'INSERT DESIRED LABEL'
  },
  INSERTYOURTYPE2: {
    label: 'INSERT DESIRED LABEL'
  }
};
```

6. Load the XML File using the XMLHttpRequest object that allows you to retrieve a file from the same domain (within the initMap() function)
    a. Copy the following code to call the XML file and parse through the location attributes



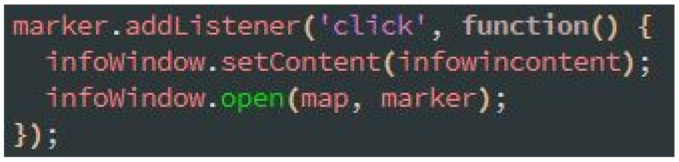
```
downloadUrl('http://INSERT_YOUR_PROJECT_IP_ADDRESS/INSERT_YOUR_PHP_FILE_NAME', function(data) {
var xml = data.responseXML;
var markers = xml.documentElement.getElementsByTagName('marker');
Array.prototype.forEach.call(markers, function(markerElem) {
  var id = markerElem.getAttribute('id');
  var name = markerElem.getAttribute('name');
  var address = markerElem.getAttribute('address');
  var type = markerElem.getAttribute('type');
  var point = new google.maps.LatLng(
      parseFloat(markerElem.getAttribute('lat')),
      parseFloat(markerElem.getAttribute('lng')));

  var infowincontent = document.createElement('div');
  var strong = document.createElement('strong');
  strong.textContent = name
  infowincontent.appendChild(strong);
  infowincontent.appendChild(document.createElement('br'));

  var text = document.createElement('text');
  text.textContent = address
  infowincontent.appendChild(text);
  var icon = customLabel[type] || {};
  var marker = new google.maps.Marker({
    map: map,
    position: point,
    label: icon.label
  });
```

b. Create the downloadUrl() function to load the file (make it the last thing inside the <script> tags, outside the initMap() function)



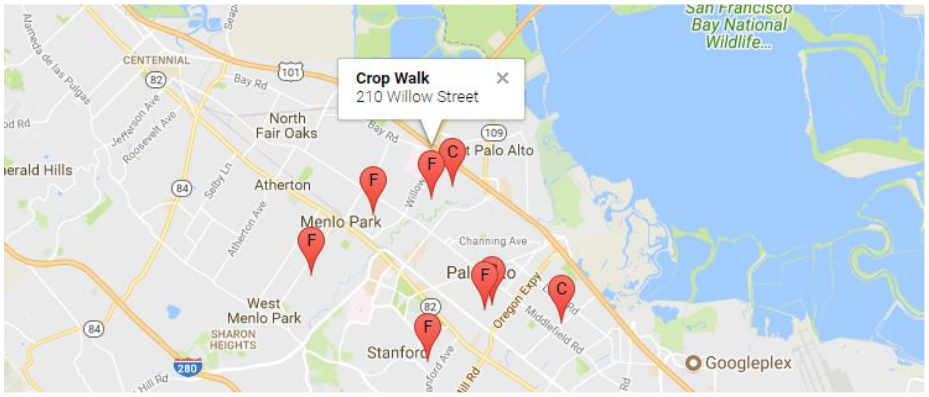
```
function downloadUrl(url, callback) {
  var request = window.ActiveXObject ?
      new ActiveXObject('Microsoft.XMLHTTP') :
      new XMLHttpRequest;

  request.onreadystatechange = function() {
    if (request.readyState == 4) {
      request.onreadystatechange = doNothing;
      callback(request, request.status);
    }
  };

  request.open('GET', url, true);
  request.send(null);
}
```

7. Create markers and information windows (within the initMap() function)



```
marker.addListener('click', function() {
  infoWindow.setContent(infowincontent);
  infoWindow.open(map, marker);
});
```

8. Run and test code
    a. Use PhoneGap or your desired text editor to preview the map in a browser/application viewer (see “General Setup” for PhoneGap instructions)
    b. Look for the markers corresponding to the locations in your MySQL database and click on them to test that the information windows work correctly


Crop Walk
210 Willow Street
North Fair Oaks
Atherton
Menlo Park
West Menlo Park
Palo Alto
Stanford
Googleplex
San Francisco Bay National Wildlife...