learntoprogram.tv

Lab 3: Obtaining and Mapping The User's Location

In this lab you will use the location feature to create a map centered around the user's location. You will learn:

1. To use the browser to access the user's location.
2. To use Javascript to create features based on the location.
3. To use external APIs to add functionality to your website (in particular, the google maps API).

1. Let's get started with a basic mobile friendly HTML page. Include jQuery Mobile, the Viewport meta tag, and a 'page' div with header, a footer, and a main section. Fix the footer to the bottom of the screen.

```
1.  <!DOCTYPE html>
2. <html>
3.   <head>
4.     <title>
5.       LAB 3
6.     </title>
7.     <meta name="viewport" content="width=device-width, initial-scale=1" />
8.     <!-- jQuery Mobile include tags here… -->
9.     <style>
10.      #footer{
11.        position: fixed;
12.        left: 0px;
13.        bottom: 0px;
14.        width: 100%;
15.      }
16.    </style>
17.  </head>
18.  <body>
19.    <div data-role="page">
20.      <div data-role="header"><h1>Mobile Map</h1></div>
21.      <div data-role="main"></div>
22.      <div id="footer" data-role="footer"><h1>Lab 3</h1></div>
23.    </div>
24.  </body>
25. </html>
```

Next, write some Javascript to detect the location and log it to the console.

2. Add a script tag in the head of the HTML document.

```
1.     <script>
2.     </script>
3. </head>
```

3. Assign a function to window.onload. This function will run once the page has been loaded.

```
1. <script>
2.     window.onload = function(){
3.     }
```

4. In this 'window.onload' function, write an if statement to check if the geolocation feature is available on the device. If geolocation is not available, the user should receive an alert that the location is not available.

```
1.     window.onload = function(){
2.         if (navigator.geolocation) {

3.         } else {
4.           alert("location not available");
5.         }
```

5. If the geolocation is available, the script should retrieve the current position of the device using the 'getCurrentPosition' function of the geolocation object.

```
1.  window.onload = function(){
2.    if (navigator.geolocation) {
3.      navigator.geolocation.getCurrent
    Position();
4.    } else {
5.      alert("location not available");

6.    }
```

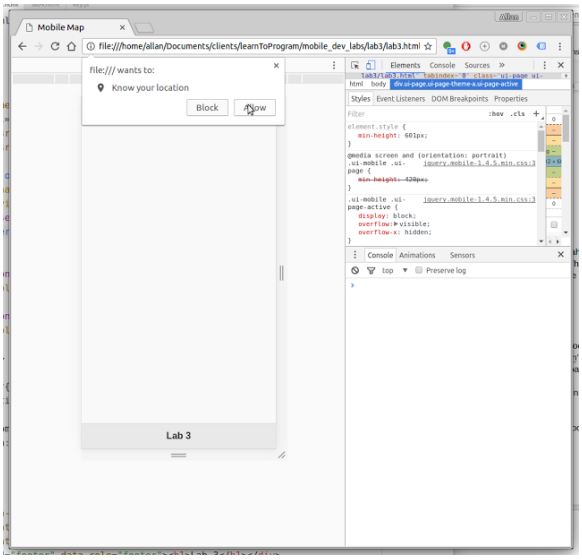
6. The 'getCurrentPosition' function should accept two callback functions as parameters. The first callback function, called 'onPosition' will get called if the 'getCurrentPosition' function is successful. The second, called 'onError' will get called if the 'getCurrentPosition' function produces an error. You will create the callback functions in the next step, here just add the callback functions as parameters to the 'getCurrentPosition' function.

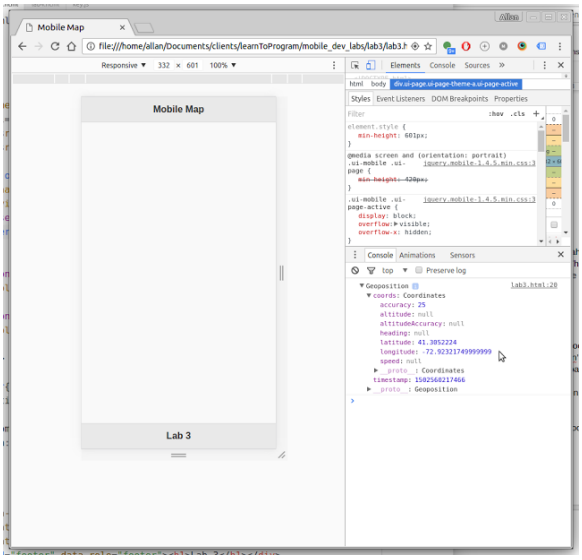
```
1.      window.onload = function(){
2.    if (navigator.geolocation) {
3.      navigator.geolocation.getCurrent
    Position( onPosition, onError );
4.    } else {
5.      alert("location not available");

6.    }
```

7. Create the 'onPosition' and and 'onError' functions. The 'onPosition' function should take a position object as a parameter and log it to the console. The 'onError' function should take a error object as a parameter and log the error message to the console.

```
1.       window.onload = function(){
2.
3.  }
4.  function onPosition(position){
5.    console.log(position);
6.  }
7.  function onError(error){
8.    console.log(error.message);
9.  }
```

8. In your browser, open the HTML file using the mobile emulator. If prompted to allow your location, go ahead and accept. If all goes well, you should see the location object printed to the console output.





Challenge Yourself:

1. Simulate different user locations then your currrent location in your browser using the developer tools.
2. Place text with the latidude and longitude coordinates of the location on your site when the broswer loads the position object.

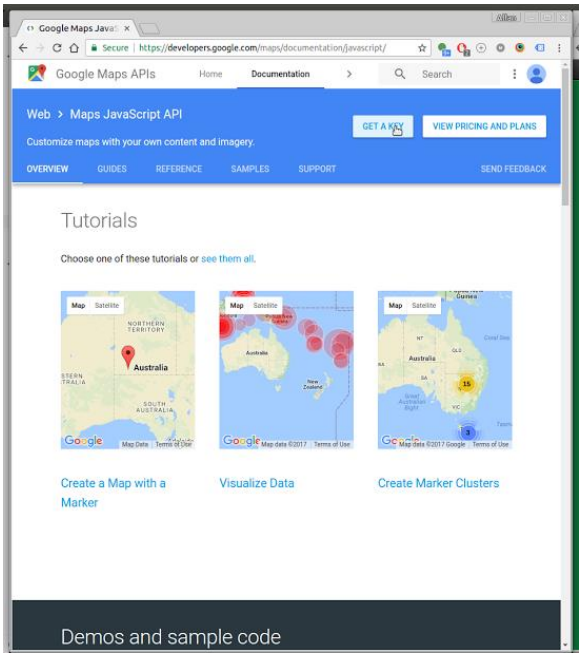
Now that you've obtained the device position, let's use it to render a map centered around the device location.

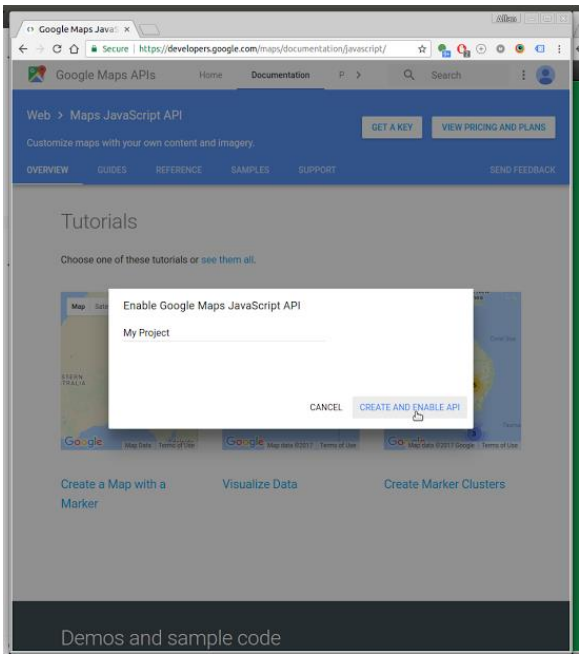
First, you will need to obtain a key for the google maps API.

9. If you do not have a google account yet, create one first. Log in to your google account and go to the Javascript documentation page:

https://developers.google.com/maps/documentation/javascript/

10. Click "Get a Key" and in the following window, select a project or create a new one. Click the button to "Enable API". A key should show up, copy this key and save it somewhere safe. In practice you should treat your keys like passwords and not share them with anyone.





11. In the same directory as the main HTML file, create a new file called key.js where the key will be stored. Write the following line in the file and place the key you just obtained in the place of the comment. Save the file.

```
1.  key = // "your key here"
```

12. Add a script tag just before your geolocation script to read in the Javascript file you just created.

```
<script type="text/javascript" src="key.js"></script>
<script>

</script>
</heaad>
```

Now that you have a key, let's build the map feature into the HTML.

13. Create an iframe inside the main section with the attributes listed in the following example. Give the iframe an id of 'map'. This iframe will be used by the google maps API to render the map.

```
    <div data-role="header"><h1>LAB 3</h1></div>
      <div data-role="main">
        <iframe id="map" width="100%" height="300px" frameborder="0" style="border:0;background-color:lightblue;"></iframe>
      </div>
```

14. In the 'onPosition' function, save the longitude and latitude to variables.

```
function onPosition(Position){
  console.log(Position);
  var lat = position.coords.latitude

  var long = position.coords.longitude
}
```

15. In the 'onPosition' function, use the latitude and longitude along with the API key to build a url string. You will use this url string to query the google maps API.

```
<script type="text/javascript" src="key.js"></script>

 <script>

   function onPosition(Position){
     console.log(Position);
     var lat = position.coords.latitude
     var long = position.coords.longitude
     var url = "https://www.google.com/maps/embed/v1/place?key="
     url+= key+"&q="+lat+","+long
   }
```

16. In the onPosition function, set the src of the 'map' iframe to the url string that you just created.

```

function onPosition(Position){
  console.log(Position);
  var lat = position.coords.latitude

  var long = position.coords.longitude
  var url = "https://www.google.com/maps/embed/v1/place?key="
  url+= key+"&q="+lat+","+long
  document.getElementById("map").setAttribute("src",url)
}
```

17. Now, let's go ahead and test out the map. Open up your HTML file in your browser. If everything worked properly, you should see a map centered around your current location.

[graphic 7]

Challenge Yourself:

-Create a site that uses location information to find and list local restaurants.