Session 7 - Geolocation API

We will cover:
1) Collecting coordinates from the gps
2) Watching a gps position and updating as the user moves.
3) The various pitfalls that can cause problems when using geolocation api.

Step 7a - Accessing the gps

The geolocation api is a really interesting api. It shows us where the web is heading. Web pages are going to be able to access increasingly greater amounts of hardware. Allowing us web developers to develop way outside of the confines of the traditional web page environment.

The api in itself is very simple. There are two main functions `getCurrentPosition` and `watchPosition.`

`getCurrentPosition:`	Triggers a callback once
`watchPosition:`	Starts watching the gps and triggers a callback everytime it has a new position (this can be very frequently when outside and the gps is active).

Both functions can be passed the same parameters and work in almost exactly the same way.

1) Open session7/step7a.html. You will notice I have created two functions. One for processing our gps output and one for processing any errors.

2) Now lets wire these functions to our position with the geolocation api. Under the last function trigger a one shot getCurrentPosition. An example of how you trigger it is given below.(Note you'll have to change it a bit).

```
navigator.geolocation.getCurrentPosition(functionTriggeredOnNewPosition, functionTrig-
geredOnError, {enableHighAccuracy:true});
```

The first time you successfully run the app your browser should ask you if it's ok for the website to have access to your position. If you choose no. Our error function will be triggered.

You'll probably notice that accuracy is relatively low and several of the properties are null. This is due to the fact that the gps requires the user to have a view of the sky, the more sky you can see the better your gps accuracy can be. As your working indoors, and probably can't see any sky, your mobile device will be using a different method for calculating your position. Such as wifi mapping.

3) We'll now replace our getCurrentPosition with a watchPosition. As the two methods take the same parameters and return the same output you can simply change function names.

Now your gps will start watching your position. If you wan't to you can now move to an area where you can see the sky, to see if you can get a better accuracy reading and some extra information such as altitude.

When working with the watchPosition you should never use it as I have just shown. You should assign the watch to a variable. This is because the browser will watch the gps position until the user leaves the page. Not good from a battery perspective, the gps is one of the most power hungry components in your phone. By assigning the watch to a variable you can trigger a clearWatch when you no longer require the gps.

You can achieve this as follows.

```
var watchId = navigator.geolocation.watchPosition(functionTriggeredOnNewPosition,
functionTriggeredOnError, {enableHighAccuracy:true});

function buttonClickHandler() {

      // Cancel the updates when the user clicks a button.
      navigator.geolocation.clearWatch(watchId);
}
```

Reference:

http://dev.w3.org/geo/api/spec-source-v2.html

If time permits have a look at ReverseGeocode.html . It's quite an interesting google service. Note due to google maps key constraints the example will only work on local-host i.e. in your desktop browser.

Step 7b - Mapping our position

So we can get the coordinates from the gps. That’s not really that helpful unless we can combine it with other data such as a map. In this example you’ll wire up a static map from openstreetmap to show your location.

1) Open session7/step7b.html look at the code, try and understand what it does. Run the code in you browser.

2) Your task is to make the map center on your location with a map pin for extra measure.

3) Try changing the size of the generated map to your browser window size (google if you don’t know the javascript command. Hint it is window. something !!!)

4) Can you alter the map parameters so you zoom closer to the ground?

Step 7c - retrieving data based on location

Our wikipedia app. illustrates how you can query external services based on text input. There are also a number of services you can query based on location. There is in fact an unofficial wikipedia geoapi. It's a little slow but it shows how you can use location to filter data.

Checkout the api documentation if your interested here:

http://wikilocation.org/documentation/

In this example we've added a panel to our wikipedia app. to get article titles that are nearby. As soon as the the user opens the panel we'll trigger a `navigator.geolocation.getCurrentPosition`

1) Look at the code for step7c.html . Scroll down until you find the code that is triggered when our local panel is set to show():

```
            localResultsPanel.show = function() {
                  localResultsPanel.getDomElement().style.display = "-webkit-box";
                  localResultsPanel.wikiListPanel.mainContent.setText("Loading...");
                  /*
                   *
                   * Exercise 7c here
                   *
                   *
                   *
                   */

                  localResultsPanel.geoload(51.500688,-0.124411, 10);
              };
```

2) Checkout the page in your browser and click on the local menu item. It all works, but we are not in London.

3) Add some geolocation magic so that when you click on the local menu item you receive info on wikipedia articles near you.

Congratulations you've just built your first location based service :-)

Reference
http://wikilocation.org/documentation/