**Web and Mobile Apps and Programming – Tracking Location**

HTML 5 offers in built functionality to track a users' location.  This works particularly well on a mobile phone, where you can take advantage of GNSS and mobile network positioning.  It is less useful on a more fixed device - e.g. a laptop - as these don't have in-built GNSS, so the tracking code makes use of the IP address to provide an approximate location.

Before you start this practical:

1. Pull/merge the code from the *leaflet* branch into the *main*branch on GitHub.

2. Create a new branch called *locationservices*

**NB:  Do NOT delete the *leaflet* branch as we will use this for assignment marking**

**Step 1 - Exploring Bootstrap's Responsive Mode Menus**

Bootstrap provides [responsive-mode](https://www.w3schools.com/css/css_rwd_intro.asp#:~:text=Web%20pages%20can%20be%20viewed,use%2C%20regardless%20of%20the%20device.&text=It%20is%20called%20responsive%20web,look%20good%20on%20any%20screen.) web pages, which means that items can be set to only appear on screens of a specific size.  This allows you to customise your web page and show the user different options depending on the device they are using.

The index.html file provided earlier in the module has a sub menu called *Screen Size Check* which contains HTML menu option set up to show different screen sizes.  Start your web server and your APP and test these out   (the instructions are for the Chrome browser)

1. Click on the three dots at the top right of the screen, select *More Tools* and *Developer Tools* to launch the developer tools option

2. Now click on the small icon immediately to the left of the *Elements* option - this will toggle the *Device Toolbar* (you can also press ctrl-shift-m)

A screenshot of a map

Description automatically generated

3. You can adjust the pixel values at the top of the screen to see what happens to the menu options at [different bootstrap screen sizes](https://getbootstrap.com/docs/5.0/layout/breakpoints/).

A map of a city

Description automatically generated

**Step 2 - Set up location tracking menu options**

NB: We only want location based services to be active on mobile devices, i.e. on smaller screens -  Bootstrap [MEDIUM  or lower size](https://getbootstrap.com/docs/5.0/layout/breakpoints/).

1. Add a new JavaScript file called *locationServices.js -* include "use strict" and reference this file from your *index.html* file

2. In the new file, add two functions with alert messaages *swtichLocationServicesOn* and *switchLocationServicesOff*

3. Create a new menu group called *Location Tracking*that only appears when the screen size is medium or below.

4. The code to create the varying visibility options is as follows

<!-- menus that change with scren size -->

<div class="nav-item dropdown d-sm-down">

<a href="#" class="nav-link dropdown-toggle" data-bs-toggle="dropdown"><i class="fa fa-laptop me-2"></i>Screen Size Check</a>

<div class="dropdown-menu bg-transparent border-0">

<!-- only display when screen width is medium -->

<!-- default with block is that everything above is also visible so need to use lg-none to stop that -->

<a class="dropdown-item d-none d-md-block d-lg-none" href="#" onclick="alert('d-none d-md-block d-lg-none')"></i>Only medium</a>

<!-- only display when screen width is medium or below -->

<!-- when you use none, everything below is VISIBLE, everything above is not -->

<a class="dropdown-item d-lg-none" href="#" onclick="alert('d-lg-none')"></i>Medium or below</a>

<!-- only display when screen width is medium or above -->

<!-- d-xs-none does not exist, use d-none -->

<a class="dropdown-item d-sm-none d-none d-md-block" href="#" onclick="alert('d-sm-none d-none d-md-block')"></i>Medium or above</a>

<!-- only display when screen width is NOT medium -->

<!-- default when using none is that everything below is visible, everything above is not

so in this case you also need to say specifically that everything above is visible using d-lg-block -->

<a class="dropdown-item d-md-none d-lg-block" id="notmedium" href="#" onclick="alert('d-md-none d-lg-block')"></i>Not medium</a>

<!-- only display when screen width is large or above -->

<a class="dropdown-item d-sm-none d-md-none d-none d-lg-block" href="#" onclick="alert('dropdown-item d-sm-none d-md-none d-none d-lg-block')"></i>Large or above</a>

5. Basing your work on the above example, add two sub menus *Swtich Location Tracking On* and *Switch Location Tracking Off*

6.  Note that we'll use a call with a parameter to switchLocationTrackingOn - this is because we want to have different options in future for what happens when location tracking is enabled.   Your code should look like this:

<a class="dropdown-item d-lg-none" href="#" onclick="switchLocationServicesOn('showPosition')"></i>Switch Location Tracking On</a>

This passes a string parameter which is the name of the function that we want to call when location tracking is activated.  By changing the parameter, we can change what happens.

**Step 3 - Set up location tracking menu options**

1. The first thing that we need to do is create a global variable to hold the location tracker - that way we can retrieve it when we want to switch tracking off.  At the top of your *locationServices.js* code (underneath "use strict") add the following:

/\*\*

\* locationTrackerId

\*

\* stores the ID of the location tracker so that it can be switched off if necessary

\* avoids having multipole location services running at the same time

\*

\*/

let locationTrackerId;

We can then add the code to track location.  To make this more robust, we should check whether the device supports location tracking and alert the user if not.  We should also check if there is an existing tracking process and remove it - we should only be running one tracking process at a time.

2. Add this code to the *swtichLocationServicesOn* function

/\*\*

\* @function swtichLocationServicesOn

\*

\*

\* @param locationFunction string - the function that is called when the locaiton tracking is swtiched on

\*

\* @description start location services - make sure that there is only one location service running at a time

\* <BR><br>having locationFunction as a parameter means that we can vary what happens with the tracking information

\*

\*

\*/

function switchLocationServicesOn(locationFunction) {

// we need to check if the location service is already in use, and if it is remove it before creating a new one

// so that we don’t have multiple tracking going on

if (navigator.geolocation) {

try {

(navigator.geolocation.clearWatch(locationTrackerId));

}

catch (e){

console.log(e);

}

// need to tell the tracker what we will do with the coordinates – showPosition

// also what we will do if there is an error – errorPosition

// also set some parameters – e.g how often to renew, what timeout to set

// timeout - how long the system will keep trying for a location before rasing an error

// maximumAge - how long to cache the position

const options = {

enableHighAccuracy: true ,

maximumAge: 5000,

timeout: 27000

};

// start the new location tracking service

// showPosition is the function that processes location information once it is obtained

// errorPosition is the function that is called if there is an error

// options are the settings above - height accuracy, how frequneelty position is measured

// we use window[locationFunction] here as we receive the function name as a STRING from the menu but need to CAST (convert) it to a function call (strings are just text)

locationTrackerId = navigator.geolocation.watchPosition(window[locationFunction],errorPosition, options);

}

else {

alert("Location tracking not supported on this device");

}

}

The *watchPosition*function that is used to swich location tracking on requires three parameters  - the function that processes the location information, the function that handled any errors, and the options for tracking (e.g. whether high accuracy tracking is required).

The name of the function to call to process the location information has been provided as a *string* parameter called *locationFunction.* However, JavaScript doesn't automatically take a string and turn it into a function call - they're two different things.   So we need to use the window[xxx] function to call the function.

We now need to write the two functions to enable tracking.

3. Add the function that processes the location information.  For now, we'll just put the coordinates and other information into the DIV we used in a previous practical for the map click.

/\*\*

\* @function showPosition

\*

\* @description update the user's location on the web page

\*

\* @params position - the location information, derived from the watchPosition command

\*

\*/

function showPosition(position) {

console.log("you have moved");

document.getElementById('clickCoordinates').innerHTML = " Lat: " + position.coords.latitude + " Lng: " + position.coords.longitude +" Horizontal Accuracy: "+ position.coords.accuracy + " Altitude Accuracy: "+position.coords.altitudeAccuracy+" Heading: "+position.coords.heading+" Speed: "+position.coords.speed+" Altitude: "+position.coords.altitude;

}

4. For error handling, we'll try an alert message.

/\*\*

\* function errorPosition

\*

\* if there is an error in the location tracking service, run this code

\*

\*/

function errorPosition(error){

console.log(error);

}

5. To complete the practical, upload your code, test, commit and push to GitHub.  You can test on your mobile phone or in a browser - and you may get a warning that the App wants to track your location (this is for security reasons, so that an App user knows if they are being tracked).

A screenshot of a phone

Description automatically generated

If you test in a  browser, you can add 'fake' locations via the sensors option (in Chrome).  Instructions are here:  [sensors](https://developer.chrome.com/docs/devtools/sensors) .   If you set the location to *other* you can change the location and see the DIV contents change.

A screenshot of a computer

Description automatically generated

**Step 4 - Switch Location Tracking Off**

1. Add the following code to the *switchLocationServicesOff* function.

// make sure to ues the locationTrackerId variable so that the tracker is the one that was created

// use a try/catch statement just in case locationTrackerId has not yet been used (i.e. it would be undefined and cause an error)

try {

navigator.geolocation.clearWatch(locationTrackerId);

}

catch (error){

// no need to do anything if there is an error

}

// clear the DIV that contains the tracking information

document.getElementById('clickCoordinates').innerHTML = "";

This switches off the tracking function and also clears the HTML DIV

2. Upload your code and test on your laptop/PC and also on your mobile phone.

3. Once the code is working, commit and push to GitHub. Don't forget to check that the code is uploaded to GitHub.