CSCU9B2 Practical 8: Location-Aware Web Pages

NOT USED (DOES NOT ALL WORK AS ADVERTISED)

Aims:

- **To use JavaScript to make use of location information.**
- **This practical is really for those who need a little more challenge.**
- **This sheet DOES NOT CONTAIN a checkpoint.**

Please register your practical attendance: Go to the GROUPS\CSCU9B2 folder in your Computer folder and double-click on the Register icon. Ask a demonstrator if you need help or something goes wrong.

JavaScript and Location-aware Web Pages.

As more and more devices which use the Internet are portable, and more and more users want to use the WWW to find out about the area that they are currently in, interest has grown in *location-aware* web capabilities. That is, capabilities that use the location of the user's web browser. Some devices are fitted with specific hardware for determining their location (for example using Global Positioning System (GPS) technology), whereas others can use the mobile telephone system or local wireless networks to determine their location. And static computers can be located using their Internet Protocol (IP) address. Of course, if a portable device does not have GPS, and is out of reach of mobile phone and wireless networks, it cannot help to determine its position. But such situations are relatively rare.

One can check whether a browser has the capability to support location at all: if `navigator.geolocation` evaluates to `false` then the browser does not support location. Checking `navigator.geolocation` only checks whether the browser supports geolocation. When you actually start to use the geolocation services, the browser will ask you whether you wish to give that particular web page permission to use them: if this is refused, calls to geolocation services will not work.

Example 1: Simply using the services provided.

We want to create a web page that will report the latitude and longitude of the browser. To do this, we need first to check if geolocation services are provided, and then to use them, to find out the location of the user.

The particular service we will use is called:
`navigator.geolocation.getCurrentPosition()`

This function takes two parameters: these parameters are functions. What happens is that the call is made, and if it is successful, the first function is called, and if it is unsuccessful, the second function is

called. So the first function should display the position (or do something with it), and the second function should display some sort of an error message. It is allowed just to supply the first function.

If the first function is called, it is passed a parameter which contains a `Geoposition` object (which contains `coords` which in turn contain the `latitude` and `longitude` values).

This is probably easier to understand by using it than by explaining it! Imagine that we have a function `showPosition` which takes a parameter `position`, which is a `Geoposition` object:

```
function showPosition(position)
```

The latitude of the `Geoposition` object position is `position.coords.latitude`, and the longitude of the `Geoposition` object position is at `position.coords.longitude`. So one could have a function:

```
function showPosition(position) {
      var latitude = position.coords.latitude ;
      var longitude = position.coords.longitude ;
// now use these values in a really simple way
      alert("Your position is " + latitude + " , " + longitude) ;
}
```

Then the call that uses the geolocation services would be:

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

If it goes wrong, nothing will happen, but we'll come to that!

For now, put together a web page that implements this: put the function `showPosition` in the header of the HTML file, and call the function using this code inside a <script> tag in the body:

```
if (navigator.geolocation) {
      alert("Geolocation support") ;
      // now use the geolocation capabilities
      navigator.geolocation.getCurrentPosition(showPosition) ;
      }
      else {
      // geolocation is not supported at all
      alert("No geolocation support") ;
      }
```

Try it out: when you are asked for permission to use the geolocation services, provide it, but just for this particular run.

Of course, various things can go wrong, and best practise is to check these, and report problems to the user. To do this, we need another function, one that can tell the user about errors: we will pass

the name of this function as the second parameter to `navigator.geolocation.getCurrentPosition.`

Firstly, we need somewhere to put the error text: we can use a tag (or indeed a <p> tag or one of many others: the crucial thing is that the tag has an `id` that we can use in the function).

```
<div id ="errorcode">
Error code will go here
</div>
```

Now we can write the function (which again will go in the <head> of the HTML page):

```
function displayError(error)
{ // display suitable text for the error:
  //the error will have one of the values listed
      // where to put the error text
      var errorCode =  document.getElementById("errorcode") ;

  switch(error.code) // select appropriate code depending on the
                     //value of error.code
    {
    case error.PERMISSION_DENIED:
      // alter the text inside the div with id errorCode
       errorCode.innerHTML="User denied the request for Geolocation."
;
       break; // jump to end of switch statement
    case error.POSITION_UNAVAILABLE:
       errorCode.innerHTML="Location information is unavailable. " +
error.message ;
       break;
    case error.TIMEOUT:
       errorCode.innerHTML="The request to get user location timed
out." ;
       break;
    case error.UNKNOWN_ERROR:
       errorCode.innerHTML="An unknown error occurred " +
error.message ;
       break;
   }
}
```

The call to the geolocation services should be changed to:

```
navigator.geolocation.getCurrentPosition(showPosition,displayError);
```

Now run the code again, but this time, deny the request for access to geolocation services: you should see an appropriate error message displayed.

Before we start the next example, save the JavaScript in the header to a separate file, and call it `myjava.js` . Replace the JavaScript in the header by:

```
<script src="myjava.js" > </script>
```

It shouldn't make any difference to the operation of the code, but it does make it easier to work with!

Example 2: How far is it to … ?

Given the latitude and longitude of the browser, and the latitude and longitude of another location, it is not hard to calculate the distance between the two points. But since this is not a Maths class, we will supply the code! (In fact I am copying it myself from [1] page 180: it's also available at http://www.cs.stir.ac.uk/~lss/CSC9B2/javascript/distancecalc.js)

```
// Uses the Spherical Law of Cosines to find the distance
// between two lat/long points
//
function computeDistance(startCoords, destCoords) {
// convert degree co-ordinates to radian co-ordinates: uses a
builtin function
      var startLatRads = degreesToRadians(startCoords.latitude);
      var startLongRads = degreesToRadians(startCoords.longitude);
      var destLatRads = degreesToRadians(destCoords.latitude);
      var destLongRads = degreesToRadians(destCoords.longitude);

      var Radius = 6371; // radius of the Earth in km
      var distance = Math.acos(Math.sin(startLatRads) *
Math.sin(destLatRads) +  Math.cos(startLatRads) *
Math.cos(destLatRads) *  Math.cos(startLongRads - destLongRads)) *
Radius;

      return distance;
}

function degreesToRadians(degrees) {
      // a function which takes in a parameter with a value
      // in degrees and returns the angle in radians
      radians = (degrees * Math.PI)/180;
      return radians;
```

```
}
```

You can put the location of some other place (Edinburgh? Where you were born? Where you'd like to be living?) into your code: this can be done by simply including a declaration in your Javascript:

```
var ourCoords =  { // London!
      latitude: 51.5171,
      longitude: 0.1062
};
```

or, perhaps better by reading a latitude and longitude from a web form. You can then calculate how far your browser is from there, using the `computeDistance` function:

```
computeDistance(position.coords, ourCoords) ;
```

and then write this out using the same techniques as were used for writing out the errors earlier. How far is your browser from the North Pole?

Example 3: Drawing a map of your immediate area.

Many websites include a map of an area, round a hotel, or near a school or University. This clearly has to use the services of a site that actually has maps, and currently, one of the easiest ways to access these facilities is to make use of services supplied by and supported by Google. At this moment these are free, though there is no guarantee that they will always be free.

To make use of the capabilities provided by Google, we can use the JavaScript application programmer's interface (API) that they provide. This is straightforward: type

```
<script src="http://maps.google.com/maps/api/js?sensor=true">
</script>
```

into the header of your web page. What does this do? It allows us to use the code at `http://maps.google.com/maps/api/js` ; further it tells the Google system that we have a sensor here, and will be using our own location.

We also need to put the map somewhere: though it is possible to put a map inside a <canvas> it's complicated (challenge for those who need it: do it that way!), so we'll simply put it inside a . We need the to be a suitable size, and this can be achieved using CSS:

```
div#map {
      margin: 5px;
      width: 400px;
      height: 400px;
      border: 1px solid black;
}
```

and then we can create the `div` with the `id` of `map`:

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

which will be 400 by 400 pixels, with a margin of 5 pixels, and a 1 pixel solid black boundary. We are now in a position to get and display the map:

```
function showMap(coords)
{
// create a google latitude and longitude object
     var googleLatAndLong = new google.maps.LatLng(coords.latitude,
coords.longitude) ;
// options for map:
// could put these into a form and let the user change them
     var mapOptions =
     {
          zoom: 10 , // try different values
          center: googleLatAndLong ,
          mapTypeId: google.maps.MapTypeId.ROADMAP // other options
                                        //  are SATELLITE and
HYBRID
     } ;
     var mapDiv = document.getElementById("map") ;
     map = new google.maps.Map(mapDiv, mapOptions) ; // display map
}
```

You should be able to get a map of the area round your browser!