Mobile Application Development

Session 5 – Activities

The final product(s) for this session can be found at:
http://www.dcs.bbk.ac.uk/lo/mad/madexamples/session5/classactivities/zedlandhotels/zedland-hotels.html

1. Integrating with a Mobile Device

Open the Zedland Hotel Finder app *contact.html* page from session 4. If you did not complete session 4, use the completed version from the session 4 resources folder.

In the content section, add an <a> element. In this element, add the CSS class details required to make the link a JQM button with a phone icon.

```
<a href="" class="ui-btn ui-icon-phone →
ui-btn-icon-left ui-corner-all ui-btn-icon-notext"></a>
```

Now add in the *href* value that will make the link activate the device phone application when the button is clicked.

```
<a href="tel:+7810183041" class="ui-btn ui-icon-phone →
ui-btn-icon-left ui-corner-all ui-btn-icon-notext"></a>
```

Save your page, upload it to the server, and test it in your mobile device. Note that these features can only be tested in-situ in a mobile device with a browser that supports HTML 5 and phone capability.

Now repeat the above steps to add an email and an SMS link. We will use the same (mail) icon for both.

Email	mailto:zed@zedland.co.uk	ui-icon-mail
SMS	sms://+7551876198	ui-icon-mail

Finally, add the following postal address using standard markup, below the buttons.

72 Tower Park
Capital City
Zedland
ZL7867H

Again, save your page, upload it to the server, and test it in your mobile device.

2. Geolocation with HTML 5

HTML

Open all of your HTML pages in the Zedland Hotel App. In the <head> section, below the links to the Google CDN, jQuery and jQuery Mobile, create a link to the stylesheet we will use to style our map (*hotelmap.css*). Also, add <script> tags for the Google Maps API, and for the JavaScript file (*hotelmap.js*) we will use to access device geolocation and instantiate our map.

```
<link rel="stylesheet" type="text/css" href="hotelmap.css">
<script src="https://maps.googleapis.com/maps/api/js?v=3.exp"></script>
<script src="hotelmap.js"></script>
```

Save your files.

Next, in the primary navigation listview menu in *#home* in *zedland-hotels.html*, add an additional link to the external page where we will display our map. This page will be called *maps.html*.

```
 <li><a href="maps.html">Hotel Locations</a></li>
```

Save *zedland-hotels.html.*

Now create *maps.html.* In *maps.html*, create a new jQuery Mobile page. Give the page an *id* of *map-page*. Give the content section of the new page and id of *map-container*.

```
<div data-role="page" id="map-page">
   <div data-role="header">
      <a href="zedland-hotels.html" data-icon="home">Home</a>
       <h1>Locations</h1>
   </div>
   <div data-role="content" id="map-container">
   </div>
   <div data-role="footer" data-position="fixed">
      <div data-role="navbar" data-id="footernav">
        <ul>
           <li><a href="about.html">About</a></li>
            <li><a href="contact.html">Contact</a></li>
            <li><a href="login.html">Login</a></li>
        </ul>
      </div>
   </div>
</div>
```

In the <content> section of *map-page*, create a new element with an id of *map-canvas* (This is where the map will be displayed). Also create a <p> element with an id of *nogeolocation*. This element will be used to display a message to the user should his/her browser not support geolocation.

```
<div data-role="content" id="map-container">
  <div id="map-canvas"></div>
  <p id="nogeolocation"></p>
</div>
```

Save *maps.html.*

CSS

Create a new CSS file. Save it as hotel*map.css*.

We will use responsive web design techniques to ensure that our map displays correctly in whatever device it is displayed in. This involves setting relative width and height values in the CSS

In *maps.css*, add the following styles to dimension the map.

```
html, body, #map-page, #map-container {
height: 100%;
width: 100%;
padding: 0;
margin: 0;
}

div#map-canvas {
width: 100%;
height: 100%;
}
```

Note that setting relative values for the map canvas only will not suffice to make the map responsive. To achieve responsiveness we also need to set relative values for all the ancestor elements of #map-canvas.

Save hotelmap.css.

JavaScript

Create a new JavaScript file and save it as *hotelmap.js*. In *hotelmap.js*.

Start by adding the event handler below.

```
$(document).on("pagecontainershow", function (e, ui) {
  var page = ui.toPage[0].id;
      if( page == 'map-page' ) {
      . . .
 }
});
```

Note the use of the pagecontainershow event handler here. This event handler fires on the *to* page in a transition every time we transition from one page to another. In our case, we do not want our script to run on any transition, we only want it to run when *#map-page* is the *to* page. To do this we have to ascertain check the *to* page each time a transition is made, and check that the page we are transitioning to is our map page. This is done using the *toPage* property of the *ui* object.

There will be a full explanation of jQM events and event handlers in session 7.

The Next thing we will do when the event handler fires is to test whether the user's browser supports geolocation. If it does, we use a callback function (initialize) to create the map and its features. If it does not, we will display a 'not supported' status message in place of the map.

```
$(document).on("pagecreate","#map-page",function(e){
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(initialize);
  } else {
    documentgetElementById("nogeolocation").innerHTML = →
    "Geolocation is not supported by this browser.";
  }
});
```

The *initialize* function to create the map will do several things.

1. Use geolocation to get the user position
    a. Add a map marker for the user position
2. Add a hotel location using fixed coordinates
    a. Add a map marker to the hotel coordinates
    b. Add a popup infowindow for the hotel marker

Use Geolocation to get the User Position

Create a function called *initialize*. Give it a single argument, *position*. *position* will be retrieved from the device geolocation.

```
function initialize(position) {
}
```

We can now access the user's position and derive coordinates from it. We can then use these coordinates to create a position that can be used on the map.

```
function initialize(position) {
  var lat = position.coords.latitude;
  var lon = position.coords.longitude;
  var currentPosition = new google.maps.LatLng(lat, lon);
}
```

Now we have a location, we need a map to use it in. To create the map, we start by setting options for the map display. In this case, we set the zoom level, the focus (center) of the map and a control to display a dropdown of map type options.

```
var mapOptions = {
 zoom: 12,
 center: currentPosition,
 mapTypeControl: true,
   mapTypeControlOptions: {
    style: google.maps.MapTypeControlStyle.DROPDOWN_MENU
  },
}
```

We then reference the Google Maps API to instantiate our map. In doing so, we pass instructions as to where the map should be displayed. We also pass the map display options we created previously to the map.

```
var hotelMap = new google.maps.Map( →
document.getElementById('map-canvas'), mapOptions);
```

Save *hotelmap.js*. Upload your work (HTML, CSS, and JavaScript) to the DCSIS student web server. Open *zedland-hotels.html.* You should be able to see a map that is centred on your current geolocation. Use the Firefox or Chrome console to identify errors if the map is not displaying.

Add a Map Marker for the User Position

We now need to create a marker to show the user position on the map.

We start off by defining a marker image. The current position marker is a blue circle image. We simply link to a URL where the image is located to include this image.

```
var currentPositionImage ='http://www.dcs.bbk.ac.uk/lo/mad/ →
madexamples/session5/classexamples/branchmap/icons/current_pos.png';
```

Referencing the Google maps API, we then create the marker. In creating the marker, we also create instructions regarding where it should be positioned, what image to use, and the tooltip text that will show on mouseover.

```
var userPosition  = new google.maps.Marker({
 position: currentPosition,
 map: hotelMap,
 icon: currentPositionImage,
 title: 'You are here'
});
```

Save you page and upload it to the DCSIS server and view it in-situ in your mobile device. You should now be able to see the blue current position marker on your map.

Add a Hotel Location using Fixed Coordinates

To add a hotel location, we start by setting a set of fixed coordinates that correspond to the hotel location.

```
function initialize(position) {
  var lat = position.coords.latitude;
  var lon = position.coords.longitude;
  var currentPosition = new google.maps.LatLng(lat, lon);
  var churchillHotelPosition = new google.maps.LatLng(51.52307,-0.12426);
 . . .
}
```

We then create a marker for this location. This time we will use a standard fixed location marker, a red pin. This image will be downloaded from Google. The marker will be displayed on our map at the fixed coordinates we created.

```
var churchillHotelMarkerImage = 'http://chart.apis.google.com/ →
chart?chst=d_map_pin_letter&chld=A|FF0000|000000';
var churchillPosition = new google.maps.Marker({
   position: churchillHotelPosition,
   map: hotelMap,
   icon: churchillHotelMarkerImage,
   title: 'Churchill Hotel'
  });
```

Creating an *infowindow* for a Map Marker

When the user clicks the Churchill Hotel map marker on the map, he/she will see an infowindow. To create this feature, we first need to create the HTML and content for the infowindow.

```
var churchillHotelInfo =
'<div id="mappopup">'+
'<h4>Churchill Hotel</h4>'+
'<p>Five Star Hotel in the middle of capital city</p>' +
'<a href="http://www.dcs.bbk.ac.uk/lo/mad/madexamples/ →
session5/classactivities/zedlandhotels/churchill- →
hotel.html">Details</a> '+
'</div>';
```

This string is now passed to the infowindow object as its content.

```
  var churchillHotelInfoWindow = new google.maps.InfoWindow({
    content: churchillHotelInfo
  });
```

The final thing left to do is to create a Google Maps event listener that detects when the user has clicked the Churchill Hotel map marker.

```
google.maps.event.addListener(churchillPosition, 'click', function() {
 churchillHotelInfoWindow.open(hotelMap, churchillPosition);
});
```

Save you page and upload it to the DCSIS server and view it in-situ in your mobile device. You should now be able to see a fixed location map marker for the Churchill Hotel. When you click the map marker, you should see a pop-up infowindow. If there are any problems, debug your application using the Chrome or Firefox console.

Full code listing

3. Making Scripts Available to all Pages in an Application

Make sure all scripts and CSS files used in an application are included on every external page of the Zedland Hotel Mobile App.