Birkbeck
UNIVERSITY OF LONDON

# HTML 5 Enhancements

Mobile Application Development

***Session 5***

## HTML 5 Enhancements

- Because jQuery Mobile is written in HTML 5 and CSS 3, we can leverage the power of these technologies to enhance our mobile apps.
- This includes the ability use features such as:
  - Device integration (e.g. phone, sms, etc.)
  - Geolocation
  - Local storage
  - Advanced forms
  - Advanced multi-media

## Device Integration

- Most mobile devices allow several different modes of communication.
  - Email
  - Phone
  - SMS
  - VoIP
- HTML 5 provides means of integrating a mobile app with all of the above, as well as applications like SKYPE.

## Integrating with the Phone

- To minimize user input of phone numbers, we can code our pages so that a user can click on a phone number, or phone icon, opening the phone application, and dialing the selected number.
- To do this we use an <a> element, where the *href* value is a telephone number preceded by the prefix *tel:*.

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

- Example

## Integrating with SMS

- HTML 5 also allows us to integrate our app with a device's SMS or MMS capability.
- For SMS, we again use an <a> element, only this time the *href* value is a telephone number prefixed by the HTML 5 specific value *sms://*.

```
<a href="sms://+7551876198" class="ui-btn ui-icon-mail
ui-btn-icon-left ui-corner-all ui-btn-icon-notext"></a>
```

- Example

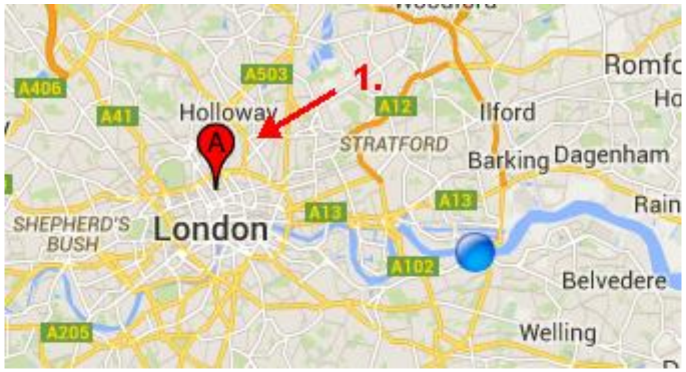
## HTML 5 Enhancement - Geolocation

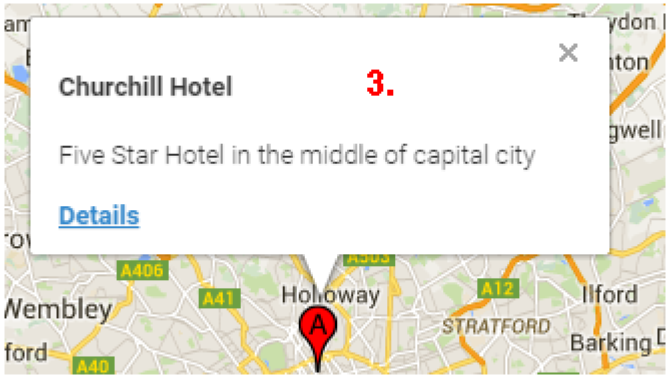
- HTML 5 comes with an API that allows us to access a device's location via its geolocation services.
- Several well known mobile apps are built with geolocation services at their heart:
  - Uber
  - Barclays Bikes
  - City mapper
  - Find my Kids
  - Tingle
- How do these apps use geolocation services?
- Can you think of any other apps that use geolocation? How do they work?

## Module Case Study App - Geolocation

- For the Zedland Hotels case study, we will use the HTML 5 geolocation in conjunction with Google Maps to help us build a hotel location feature into our app.
- This feature will allow us to find:
  - Hotels
  - Information about hotels
  - Our position in relation to hotels

## HTML 5 Geolocation

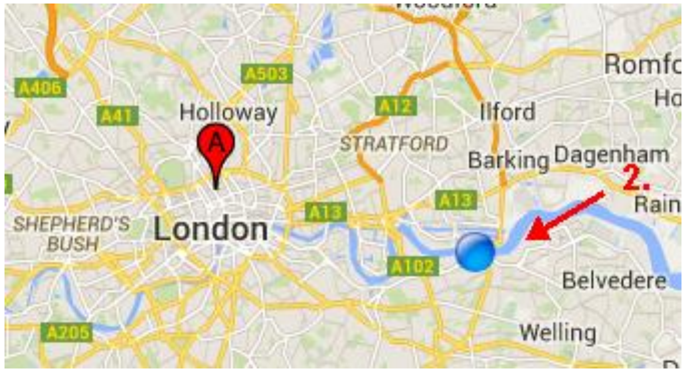




Churchill Hotel

Five Star Hotel in the middle of capital city

Details



1. *Marker* showing a hotel
2. *Marker* showing user position
3. Information about as hotel in the form of an *infowindow*.

   Example

## Displaying a Map - HTML

- The HTML markup to create a map is straightforward. We simply need to:
  - Link our pages to any CSS file that will format the map.
  - Link to the Google Maps API.
  - Link to the script used to access the device geolocation and instantiate the map.

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

  - Create the markup to display the map.
  - Create the markup to display a message if the browser does not support HTML 5 geolocation. Example.

```
<div id="map-canvas"></div>
<p id="nogeolocation"></p>
```

## Displaying a Map - CSS

- We obviously want the map to be responsive to the screen size of whatever mobile device it will be displayed in.
- To achieve this we need to set the width and height values of the map container to 100% the width and height of its parent container.
- This in itself, however,  will not make the map responsive. To do this, we need to set the width and height values of all the map container's ancestors to 100% and the padding and margin to zero. Example

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

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

## Displaying a Map – jQuery

- In normal jQuery, we invariably use the following event to run scripts on page loading.

```
$(document).ready(function() {
   //Do something
});
```

- However, because of its unique multi-page architecture, this approach to event handling does not work in jQuery Mobile.
- In jQuery Mobile, we need to use an approach to events that is capable of triggering scripts at *page*, rather than *document,* level.

# Running Scripts in jQuery Mobile

- There are several jQuery Mobile page events.
- For the purpose of this session's examples and activities, we will use the *pagecontainershow* event.
- This event is fired each time a transition to a new page occurs.
- In this case, we use *.toPage* property of the *ui* object to make certain that our script runs only when we transition to #map-page. Example.

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

## Testing for Geolocation

- Before we create our map, we first test to see if the HTML 5 geolocation feature is supported using the *geolocation* method of the *navigator* (browser) object.
- If geolocation is supported, we can then use the *getCurrentPosition* method to run the function (initialize) we will use to build our map.
- If geolocation is not supported, we display a *not supported* message to the user.

```
if (navigator.geolocation) {
  navigator.geolocation.getCurrentPosition(initialize);
  } else {
  documentgetElementById("nogeolocation").innerHTML = →
  "Geolocation is not supported by this browser.";
}
```

## Building and Displaying the Map

- The function used to build the map (*initialize*) is a JavaScript *callback function* which is called from the *getCurrentPosition* method as an argument.
- This function will perform several different tasks:
  - Get the current latitude and longitude
  - Set map options (e.g. the type of map, available controls, zoom level, map centring, etc.).
  - Apply map markers (e.g. user position and hotel positions)
  - Apply *infowindows* to map markers
  - Create the hotel map

## Getting Coordinates

- The first thing to do to create our map is to get the coordinates (latitude and longitude) of the user, and to assign coordinates to any fixed positions we wish to display.
- These coordinates are then rendered as positions that can be assigned to a google map. Example.

```
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);
```

## Specifying Map Options

- To control the way the map will display) we need to specify a set of Google Maps options.
- For our app we will specify the zoom level (higher values = greater zoom), the map center, and the inclusion of a map type dropdown control (e.g. default or satellite). Example.

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

## Building a Map

- To get the Google Maps API to construct the map, we use the API *Map* method, and pass to that method the map options we have chosen, along with the location in the HTML that the map should be rendered.

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

## Map Markers

- The next step is to draw the user’s current position on the map. This is done with a map marker. The default map marker for current position is ● .
- To set the marker we first define the location of icon to use, then create the marker, then create the properties of the marker (e.g. where it should be positioned). Example.

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

## Map Markers

- We can add markers for fixed locations to our map in the same way as we did for the user position.

```
var churchillHotelMarkerImage = URL of icon
var churchillPosition = new google.maps.Marker({
  position: churchillHotelPosition,
  map: hotelMap,
  icon: churchillHotelMarkerImage,
  title: 'Churchill Hotel'
});
```

## Adding *Infowindows*

- For each fixed location marker, we can create an associated *infowindow.*
- *An* infowindow can contain information about the location (e.g. name, description, address, URL of website, etc.).
- To create an infowindow, we start by specifying the HTML and content of the window as a string variable.

```
var churchillHotelInfo =
'<div id="mappopup">'+
'<h4>Churchill Hotel</h4>'+
'<p>Five Star Hotel in the middle of capital city</p>' +
'<a href="URL of Hotel”</a> '+
'</div>';
```

## Adding *Infowindows*

- We then create the *InfoWindow* object, passing to it the HTML and content we have created.

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

- The user will access the infowindow by clicking on its corresponding location marker. To capture this click event, we need to add the following an event listener.

```
google.maps.event.addListener(location1Marker,'click', →
function() { location1InfoWindow.open (branchMap, →
location1Marker);
});
```

- Example

## jQuery Mobile Google Maps Plugins

- As always with jQuery there are plugins for everything. This includes Google Maps.
- The most popular maps plugin is jQuery-ui-map (currently version 3).
- This plugin removes the need for Google API event listeners. Instead, you can use jQuery click events on the map and markers.
- The plugin (its own hype) is . . . *very flexible, highly customizable, lightweight (3.2kB or 3.9kB for the full version) and works out of the box with jQuery Mobile.*

## jQuery Mobile - Google Maps Plugins

- The jQuery code used by jQuery-ui-map is similar in sequence and logic to the code we saw in previous examples.
- If you have understood the earlier examples, you should be easily able to understand how the plugin works.
- In the following example (simple jQuery-ui-map) ,
  - A fixed position is created
  - The position is bound to a map
  - A marker is created
  - An info window is created
  - The info window is bound to the marker

## Reading

- Google Maps and jQuery