10

Android
The WebKit Browser

Victor Matos
Cleveland State University

Notes are based on:

Android Developers
http://developer.android.com/index.html

Google Maps Javascript API V3 Basics
http://code.google.com/apis/maps/documentation/javascript/basics.html

The Busy Coder's Guide to Android Development
by Mark L. Murphy

ISBN: 978-0-9816780-0-9

WebKit Browser

- In Android you can embed the *built-in Web browser* as a widget in your own activities, for displaying HTML material or perform Internet browsing.
- The Android browser is based on **WebKit**, the same engine that powers *Apple's Safari Web* browser.
- Android uses the **WebView** widget to host the browser's pages
- Applications using the **WebView** component must request **INTERNET** *permission*.

WebKit Browser

Browsing Power

The browser will access the Internet through whatever means are available to that specific device at the present time (WiFi, cellular network, Bluetooth-tethered phone, etc.).

The **WebKit** rendering engine used to display web pages includes methods to

1. navigate forward and backward through a history,
2. zoom in and out,
3. perform text searches,
4. load data
5. stop loading and
6. more.

WebKit Browser

Warning

In order for your Activity to access the Internet and load web pages in a *WebView*, you must add the *INTERNET* permissions to your Android Manifest file:

```
<uses-permission android:name="android.permission.INTERNET" />
```

This must be a child of the <*manifest*> element.

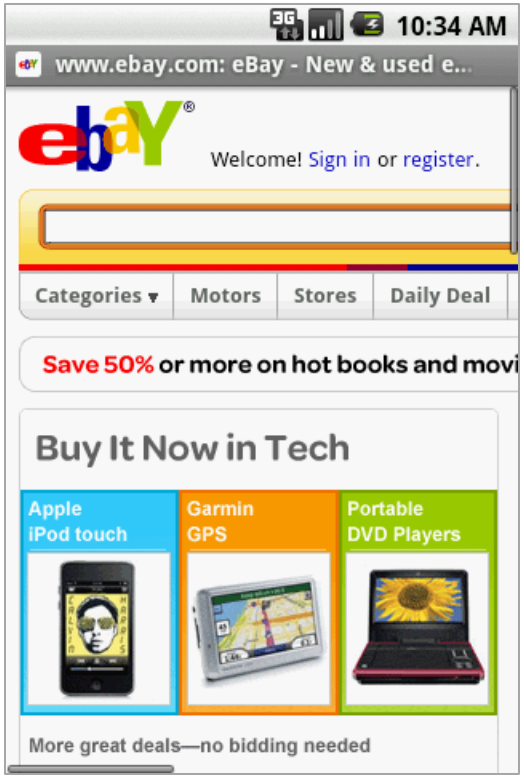
(see next example)

WebKit Browser

Example: A simple browsing experience

Let's go e-shopping

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <WebView
        android:id="@+id/webkit"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
    />
</LinearLayout>
```



WebKit Browser

Example: A simple browsing experience

Let's go e-shopping

```
package cis493.demoui;

import android.os.Bundle;
import android.app.Activity;
import android.webkit.WebView;

public class AndDemoUI extends Activity {
WebView browser;

    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);
        browser=(WebView)findViewById(R.id.webkit);
        browser.loadUrl("http://eBay.com");
        browser.getSettings().setJavaScriptEnabled(true);
    }
}
```

This app is hard-wired to eBay

WebKit Browser

Example: A simple browsing experience

Let's go e-shopping - Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="cis493.demoui" android:versionCode="1" android:versionName="1.0">

 <uses-permission android:name="android.permission.INTERNET" />

 <application android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".AndDemoUI" android:label="@string/app_name">
    <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
    </activity>
 </application>

 <uses-sdk android:minSdkVersion="3" />

</manifest>
```

WebKit Browser

Warning

If you set the URL to a site whose pages depend on *Javascript* you may see an empty, white screen.

By *default* **Javascript** is turned **off** in WebView widgets.

If you want to enable Javascript, call :

```
myWebView.setSettings().setJavaScriptEnabled(true);
```

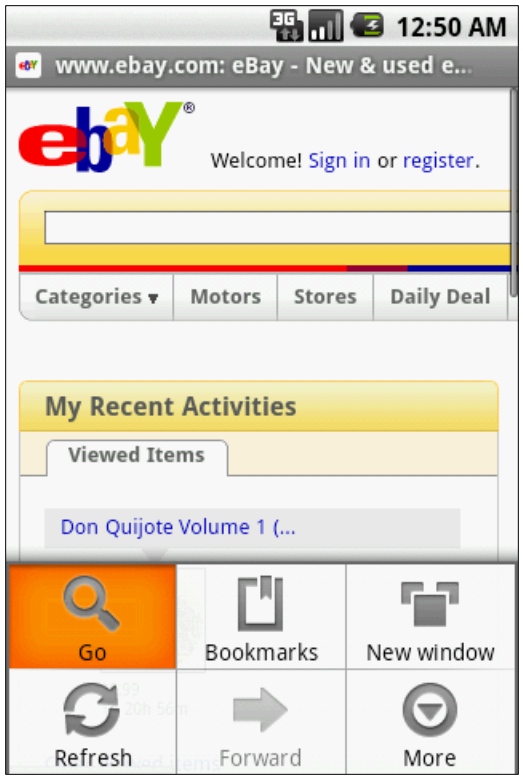
on the WebView instance.

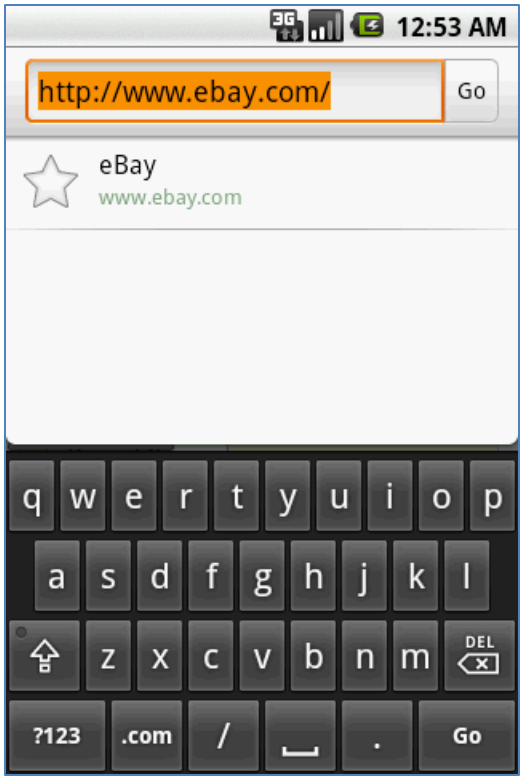
To be discussed later in this chapter.

WebKit Browser

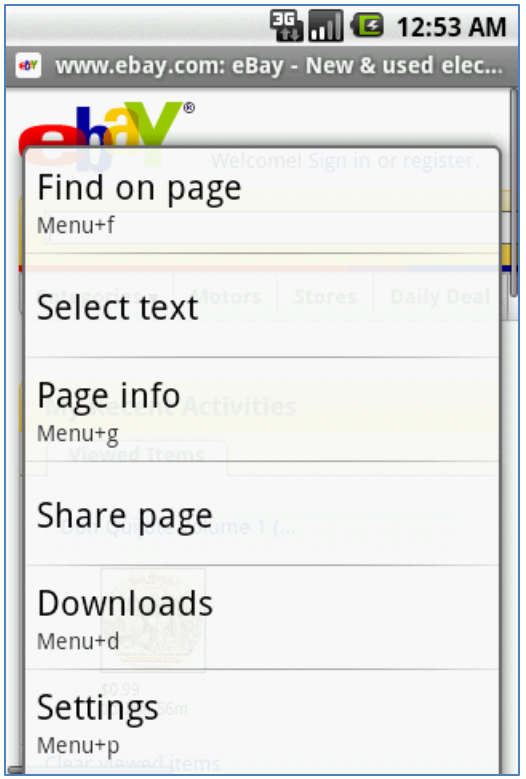
Warning

Under SDK 1.5 a WebView has a **built-in Option Menu**





Using **Go** option

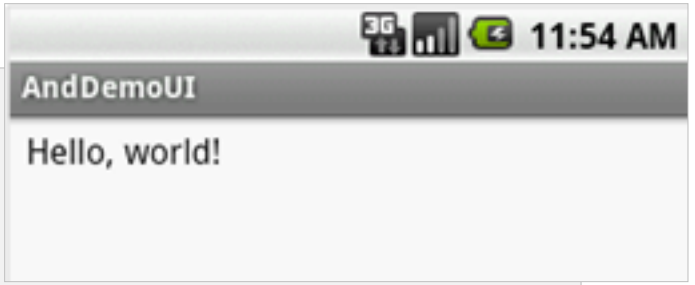


Using **More** option

WebKit Browser

Loading Data *.loadData(…)*

You may directly provide the HTML to be displayed by the browser (*a user manual for instance, or the actual app interface created as HTML instead of using the native Android UI framework*).




```
package cis493.demoui;

import android.os.Bundle;
import android.app.Activity;
import android.webkit.WebView;

public class AndDemoUI extends Activity {
     WebView browser;

     @Override
     public void onCreate(Bundle icicle) {
          super.onCreate(icicle);
          setContentView(R.layout.main);
          browser=(WebView)findViewById(R.id.webkit);

          browser.loadData("<html><body>Hello, world!</body></html>",
                           "text/html",
                           "UTF-8");
     }
}
```

Use same layout and manifest of previous example

WebKit Browser

Browser Commands

There is no navigation toolbar with the WebView widget (*saving space*).
You could supply the UI –such as a Menu– to execute the following operations:

- **reload()** to refresh the currently-viewed Web page
- **goBack()** to go back one step in the browser history, and canGoBack() to determine if there is any history to trace back
- **goForward()** to go forward one step in the browser history, and canGoForward() to determine if there is any history to go forward to
- **goBackOrForward()** to go backwards or forwards in the browser history, where *negative/positive* numbers represent a count of steps to go
- **canGoBackOrForward()** to see if the browser can go backwards or forwards the stated number of steps (following the same positive/negative convention as goBackOrForward())
- **clearCache()** to clear the browser resource cache and **clearHistory()** to clear the browsing history

WebKit Browser

Using our running example:

```
browser.goBack();
browser.goForward();
browser.goBackOrForward(-2);
browser.goBackOrForward(+2);
browser.canGoBack();
browser.canGoForward();
browser.canGoBackOrForward(-2);
browser.canGoBackOrForward(+2);
browser.clearCache(true);
browser.clearHistory();
browser.stopLoading();
```

WebKit Browser

Combining HTML + JAVASCRIPT + ANDROID

Advantages offered by Android Development

1. Access to native services on the device, including location services
2. Placement in the Android Market
3. Rapid development using the Android SDK and Eclipse.

Advantages offered by Google Maps API

1. Application exists in a server not inside a device.
2. Rapid versioning, removing the requirement for your users to download and install constant updates.
3. More frequent feature additions and bug fixes from Google.
4. Cross-platform compatibility: Using the Maps API allows you to create a single map that runs on multiple platforms.
5. Designed to load *fast* on Android and iPhone devices.

WebKit Browser

Combining HTML + JAVASCRIPT + ANDROID

Learning Strategy

- **WebView2:** Passing Objects between Android and JS
  (goal: create interconnectivity)

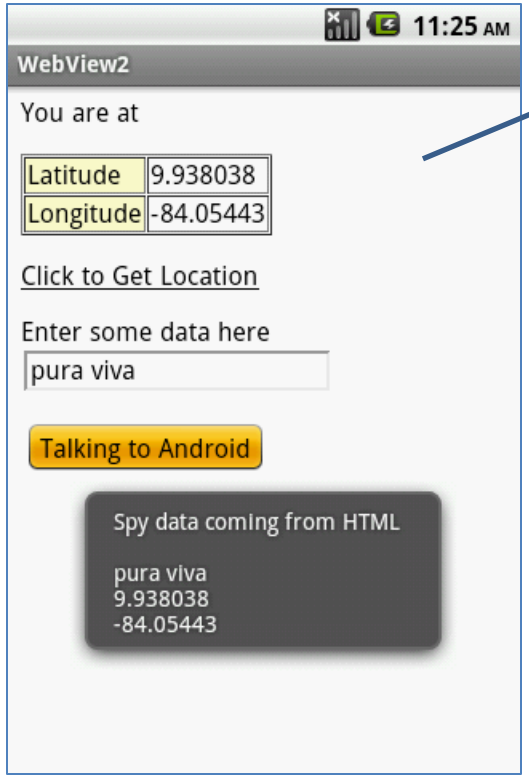
- **WebView3:** Mapping a fixed location using Google Maps V3
  (Pure HTML + JS, just update the server -no need to upgrade ALL devices carrying the application, portability, homogeneous design)

- **WebView4:** Passing a real location object to JS – draw a map centered at given location (mapping current location, combines two above).

WebKit Browser

HTML + JAVASCRIPT + ANDROID

Exchanging objects between Android & JS



HTML

Android OS

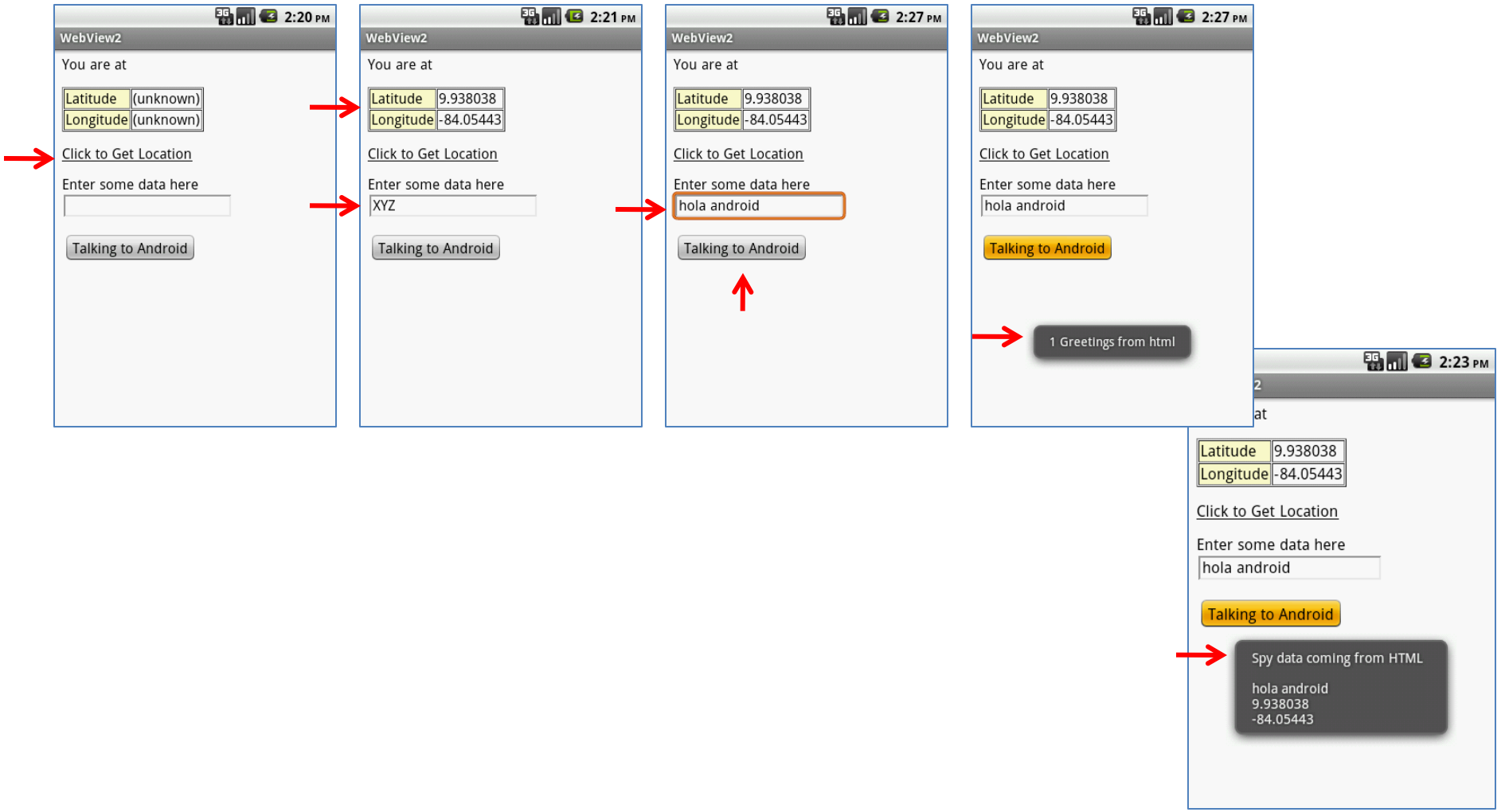
locater object

MyLocater
-spy
+getLatitude() +getLongitude() +htmlPassing2Android()

You pass an object whose **methods** you want to expose to JavaScript (*class vars not visible*)

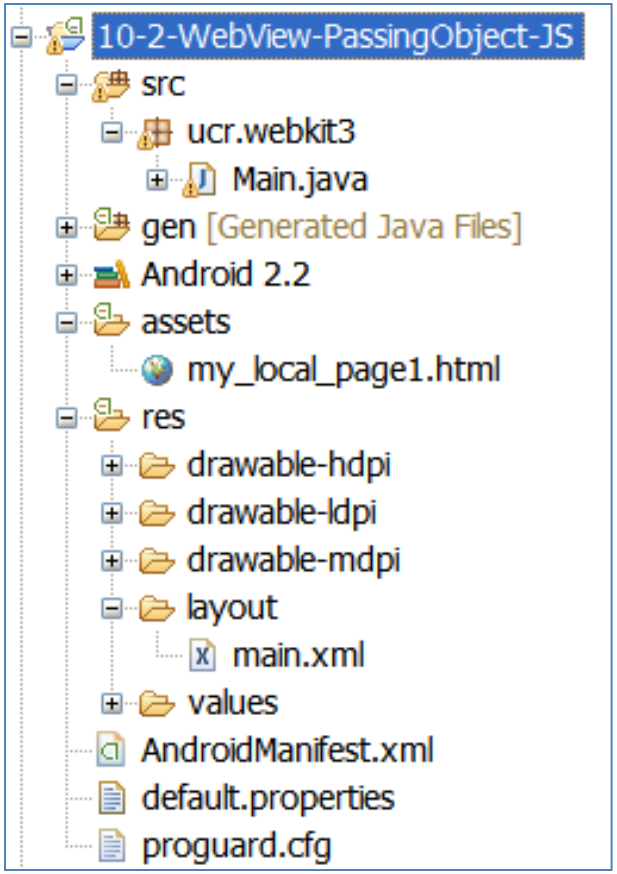
WebKit Browser

Part1. WebView2: Passing Objects between Android and JS



WebKit Browser

Part1. WebView2: Passing Objects between Android and JS



Putting the pieces together:

1. Place a **WebView** in the main.xml file
2. Place html page in the **assets** folder
3. Create the Java **object** to share with JS

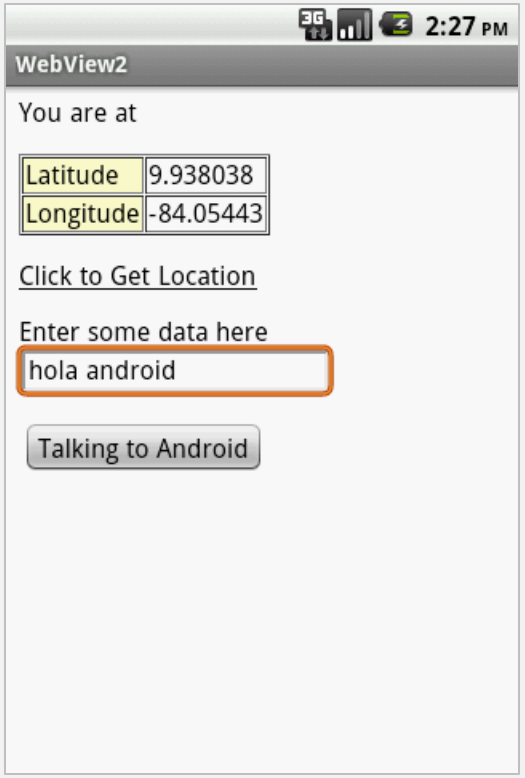
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res
/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
  <WebView
      android:id="@+id/webview"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"/>
</LinearLayout>
```

***Warning**: tested on Android 2.2*

WebKit Browser

Part1. WebView2: Passing Objects between Android and JS

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Android_Passing_HTML_JS</title> <head>
<script language="javascript">
function whereami() {
       // html asks android to provide data using object's GET methods
       document.getElementById("lat").innerHTML=locater.getLatitude();
       document.getElementById("lon").innerHTML=locater.getLongitude();
       document.getElementById("myText").value = locater.getCommonData();
}
function talkBack2Android() {
       // bridge object used to send local (html) data to android app
       locater.setCommonData("Greetings from html");
       var spyHtml  = "Spy data coming from HTML\n"
                  + "\n" + document.getElementById("myText").value
                  + "\n" + document.getElementById("lat").innerHTML
                  + "\n" + document.getElementById("lon").innerHTML;
    locater.htmlPassing2Android(spyHtml);
}
</script>
</head>

<body>
<p> You are at </p>
<table  border="1" cellspacing="1" cellpadding="1">
  <tr>
    <td bgcolor="#FFFFCC"> Latitude </td>
    <td><span id="lat"> (unknown) </span></td>
  </tr>
  <tr>
    <td bgcolor="#FFFFCC"> Longitude </td>
    <td><span id="lon"> (unknown) </span></td>
  </tr>
</table>

  <p><a onClick="whereami()"><u> Click to Get Location  </u></a></p>

  <p> Enter some data here  <input type="text" id="myText" />
  <p> <input type="button" onclick= "talkBack2Android()" value="Talking to Android">
</body>
</html>
```



WebKit Browser

Part1. WebView2: Passing Objects between Android and JS

```
public class Main extends Activity {

private WebView browser;
MyLocater locater = new MyLocater();
Location mostRecentLocation;

@Override
public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.main);

     // get a location fix (lat, lon)
     mostRecentLocation = fakeGetLocation();

     // set up the webview to show location results
     browser = (WebView) findViewById(R.id.webview);
     browser.getSettings().setJavaScriptEnabled(true);
     browser.addJavascriptInterface(locater, "locater");
     browser.loadUrl("file:///android_asset/my_local_page1.html");
}

private Location fakeGetLocation() {
     // faking the obtaining of a location object (discussed later!)
     Location fake = new Location("fake");
     fake.setLatitude(9.938038);
     fake.setLongitude(-84.054430);
     return fake;
}
```

WebKit Browser

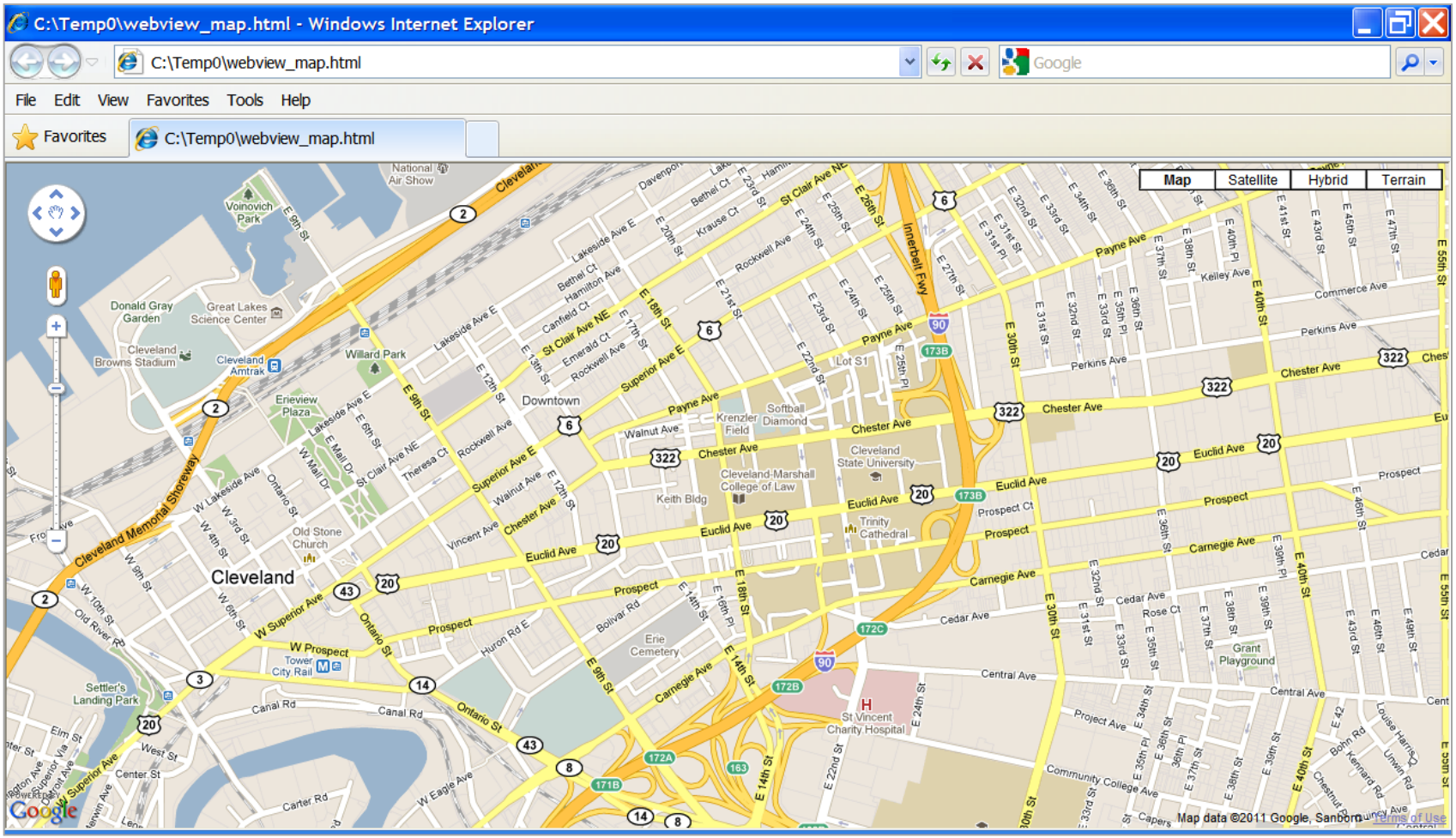
Part1. WebView2: Passing Objects between Android and JS

```
// "MyLocater" is used to pass data back and forth between Android and JS code-behind

public class MyLocater {
     private String commonData = "XYZ";

     public double getLatitude() {
          if (mostRecentLocation == null) return (0);
          else return mostRecentLocation.getLatitude();
     }

     public double getLongitude() {
          if (mostRecentLocation == null) return (0);
          else return mostRecentLocation.getLongitude();
     }

     public void htmlPassing2Android(String dataFromHtml) {
          Toast.makeText(getApplicationContext(), "1\n" + commonData, 1).show();
          commonData = dataFromHtml;
          Toast.makeText(getApplicationContext(), "2\n" + commonData, 1).show();
     }

     public String getCommonData(){
          return commonData;
     }
     public void setCommonData(String actualData){
          commonData = actualData;
     }
     }//MyLocater
}
```

WebKit Browser

Part2. WebView3: Using Google Maps V3

Webpage "webview_map.html" showing a Google map centered around Cleveland State University, Ohio (seen with IExplorer running in a Windows machine)



Link: http://code.google.com/apis/maps/documentation/javascript/basics.html

WebKit Browser

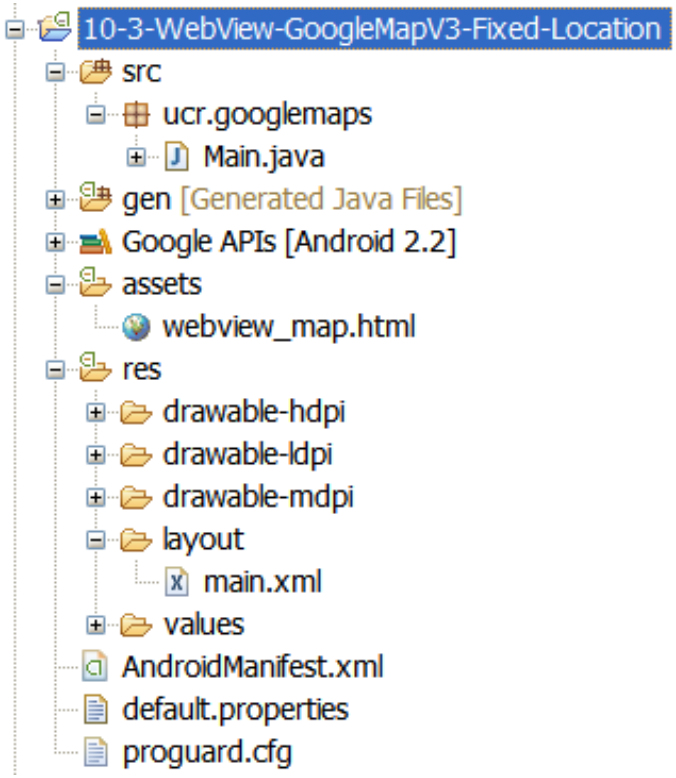
Part2. WebView3: Passing Objects between Android and JS

This is the web page: webview_map.html

```
<!DOCTYPE html>
<html>
      <head>
            <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />

            <style type="text/css">
                  html { height: 100% }
                  body { height: 100%; margin: 0px; padding: 0px }
                  #map_canvas { height: 100% }
            </style>

            <script type="text/javascript"
                        src="http://maps.google.com/maps/api/js?sensor=false">
            </script>
            <script type="text/javascript">
                    function initialize() {
                     var latlng = new google.maps.LatLng(41.5020952, -81.6789717);
                     var myOptions = {
                             zoom: 15,
                             center: latlng,
                             mapTypeId: google.maps.MapTypeId.ROADMAP
                           };
                     var map = new google.maps.Map(document.getElementById("map_canvas"), myOptions);
              }

            </script>
      </head>
      <body onload="initialize()">
            <div id="map_canvas"  style="width:100%;   height:100%"   ></div>
      </body>
</html>
```

WebKit Browser

Part2. WebView3: Porting to Android the Google Map V3 App.



Putting the pieces together:

1. Place a **WebView** in the main.xml file
2. Place html page in the **assets** folder
3. Add **permission** requests to manifest
4. Connect to Java code

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res
/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
  <WebView
      android:id="@+id/webview"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"/>
</LinearLayout>
```

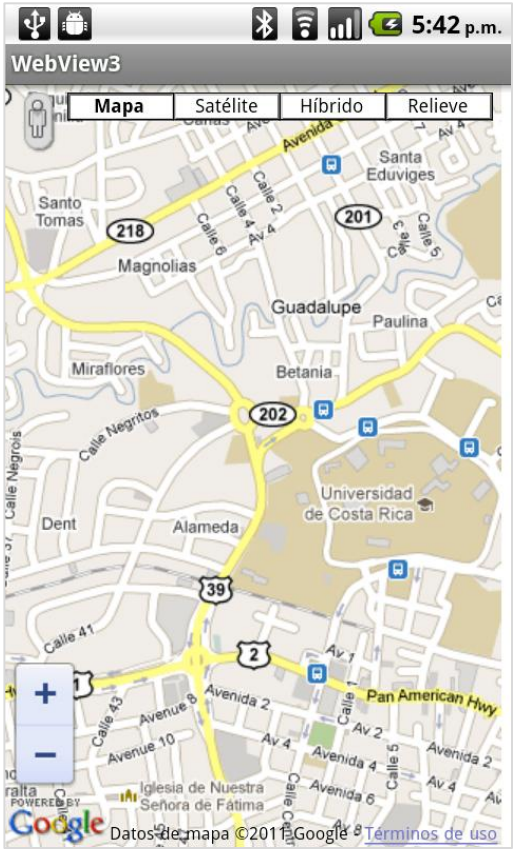
***Warning**: tested on Android 2.2*

WebKit Browser

Part2. WebView3: Porting to Android the Google Map V3 App.

Add the following permission requests to the AndroidManifest.xml file

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```



Map image shown on an Android device

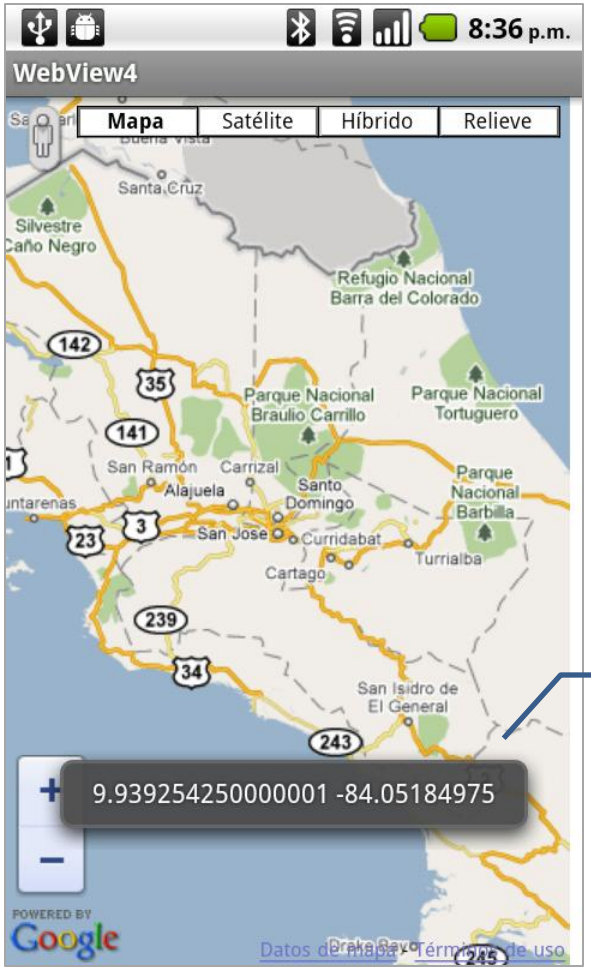
WebKit Browser

Part2. WebView3: Porting to Android the Google Map V3 App.

```
public class Main extends Activity {
  WebView browser;

  @Override
  public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);

   // connect browser to local html file showing map
   browser = (WebView) findViewById(R.id.webview);
   browser.getSettings().setJavaScriptEnabled(true);
   browser.loadUrl("file:///android_asset/webview_map.html");

  }

}
```

WebKit Browser

Part3. WebView4: Android & Google Map V3 App (real locations)



This experience combines the two previous examples:

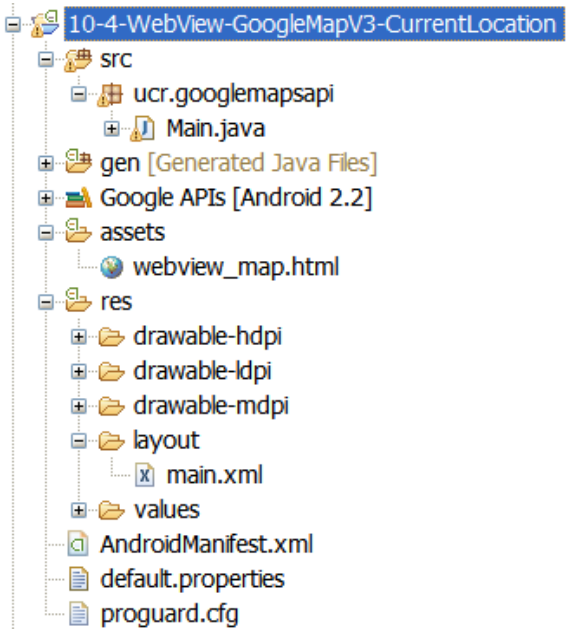
- The goal is to use an Android object to pass '*real location*' data to an html webpage.
- The page contains a JavaScript fragment to draw a map centered on the given coordinates.

Latitude and longitude detected by the device. Image taken from the Android phone.

Warning: *Make sure your target is:* ***Google APIs (API Level 8)*** *or higher.*

WebKit Browser

Part2. WebView3: Porting to Android the Google Map V3 App.



Putting the pieces together:

1. Place a **WebView** in the main.xml file
2. Place html page in the **assets** folder
3. Add **permission** requests to manifest
4. Connect to Java code

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res
/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
  <WebView
      android:id="@+id/webview"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"/>
</LinearLayout>
```

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

***Warning**: tested on Android 2.2*

WebKit Browser

Part3. WebView4: Android & Google Map V3 App (real locations)

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
<meta http-equiv="content-type" content="text/html; charset=UTF-8"/>
<title>Google Maps JavaScript API v3 Example: Marker Simple</title>

<style type="text/css">
  html { height: 100% }
  body { height: 100%; margin: 0px; padding: 0px }
  #map_canvas { height: 100% }
</style>

<script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false"></script>
<script type="text/javascript">
  function initialize() {
    //var myLatlng = new google.maps.LatLng(41.5020952, -81.6789717);
    var myLatlng = new google.maps.LatLng(locater.getLatitude(), locater.getLongitude());
    var myOptions = {
      zoom: 17,
      center: myLatlng,
      mapTypeId: google.maps.MapTypeId.ROADMAP
    }
    var map = new google.maps.Map(document.getElementById("map_canvas"), myOptions);

    var marker = new google.maps.Marker({
        position: myLatlng,
        map: map
    });
  }
</script>
</head>
<body onload="initialize()">
  <div id="map_canvas"></div>
</body>
</html>
```

Html page creates a map using 'real' coordinates passed in the 'locater' object

WebKit Browser

Part3. WebView4: Android & Google Map V3 App (real locations)

```
public class Main extends Activity implements LocationListener {
private static final String MAP_URL = "http://gmaps-
samples.googlecode.com/svn/trunk/articles-android-webmap/simple-android-map.html";
private WebView browser;
   //Location mostRecentLocation;
   LocationManager locationManager;
   MyLocater locater = new MyLocater();

@Override
protected void onDestroy() {
   super.onDestroy();
   // cut location service requests
   locationManager.removeUpdates(this);
}

private void getLocation() {
     locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
     Criteria criteria = new Criteria();
     criteria.setAccuracy(Criteria.ACCURACY_FINE); // use GPS (you must be outside)
     //criteria.setAccuracy(Criteria.ACCURACY_COARSE); // towers, wifi
     String provider = locationManager.getBestProvider(criteria, true);
     // In order to make sure the device is getting the location, request
     // updates [wakeup after changes of: 1 sec. or 0 meter]
     locationManager.requestLocationUpdates(provider, 1, 0, this);
     locater.setNewLocation(locationManager.getLastKnownLocation(provider));
}
```

WebKit Browser

Part3. WebView4: Android & Google Map V3 App (real locations)

```
@Override
public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.main);
     getLocation();
     setupbrowser();
     this.setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
}

/** Sets up the browser object and loads the URL of the page **/
private void setupbrowser() {
     final String centerURL = "javascript:centerAt("
                         + locater.getLatitude() + ","
                         + locater.getLongitude() + ")";


     // set up the browser to show location results
     browser = (WebView) findViewById(R.id.webview);
     browser.getSettings().setJavaScriptEnabled(true);
     browser.addJavascriptInterface(locater, "locater");

     browser.loadUrl("file:///android_asset/webview_map.html");
```

WebKit Browser

Part3. WebView4: Android & Google Map V3 App (real locations)

```
// Wait for the page to load then send the location information
browser.setWebViewClient(new WebViewClient() {
      @Override
      public void onPageFinished(WebView view, String url) {
      browser.loadUrl(centerURL);
      }
      });
}

@Override
public void onLocationChanged(Location location) {
      String lat = String.valueOf(location.getLatitude());
      String lon = String.valueOf(location.getLongitude());
      Toast.makeText(getApplicationContext(), lat + "\n" + lon, 1).show();
      locater.setNewLocation(location);
}
@Override
public void onProviderDisabled(String provider) {
   // needed by Interface. Not used
}
@Override
public void onProviderEnabled(String provider) {
   // needed by Interface. Not used
}
@Override
public void onStatusChanged(String provider, int status, Bundle extras) {
// needed by Interface. Not used
}
```

WebKit Browser

Part3. WebView4: Android & Google Map V3 App (real locations)

```
// ////////////////////////////////////////////////////////////////
// An object of type "MyLocater" will be used to pass data back and
// forth between the Android app and the JS code behind the html page.
// ////////////////////////////////////////////////////////////////
public class MyLocater {
     private Location mostRecentLocation;

     public void setNewLocation(Location newCoordinates){
     mostRecentLocation = newCoordinates;
     }

     public double getLatitude() {
     if (mostRecentLocation == null) return (0);
      else return mostRecentLocation.getLatitude();
     }

     public double getLongitude() {
     if (mostRecentLocation == null) return (0);
      else return mostRecentLocation.getLongitude();
     }
}// MyLocater


}//class
```

WebKit Browser

Questions ?