The Web View Widget

Introduction

- Starting with Android 4.4, the WebView is based on the Chromium project, and therefor it works very similar to the Chrome web browser.

www.chromium.org

The `WebView` Class

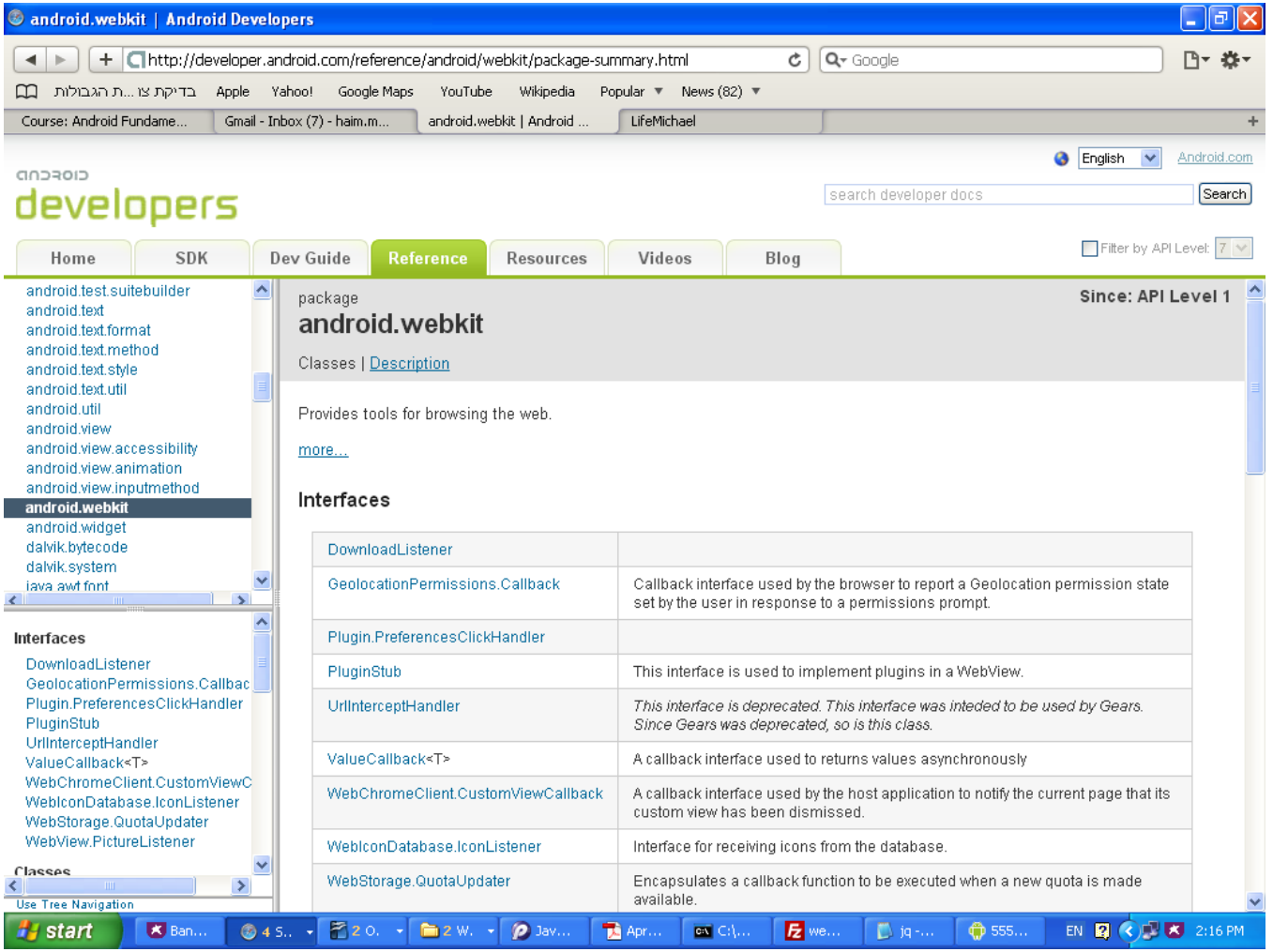
- The android platform allows us to embed the built-in web browser as a widget within the user interface of our application.
- Instantiating the `WebView` class we get an object that represents an embedded web browser.

The `android.webkit` Package

- This package includes the `WebView` class as well as many other relevant classes for interacting with the web kit browser.

http://developer.android.com/reference/android/webkit/package-summary.html

The `android.webkit` Package



The `INTERNET` Permission

- Working with the `WebView` class we might need to add the uses permission that allows accessing to the internet to the android application manifest file.

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

The `loadUrl()` Method

- Calling the `loadUrl()` method on a `WebView` object passing over a `URL` address we will get that web resource loaded within our web view object.

```
...
WebView browser =(WebView)findViewById(R.id.webby);
browser.loadUrl("http://www.lifemichael.com");
...
```

The `loadUrl()` Method

```
package com.abelski.samples;

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

public class WebViewSampleActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(bndl);
        setContentView(R.layout.main);
        WebView browser =(WebView)findViewById(R.id.webby);
        browser.loadUrl("http://www.lifemichael.com");
    }
}
```

The `loadUrl()` Method

```
<?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/webby"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />

</LinearLayout>
```

The `loadUrl()` Method

Java Script Support

- By default, the JavaScript support of the `WebView` object we are working with is turned off.
- In order to turn on the web view support for the JavaScript language we should call the `setJavaScriptEnabled()` method.

```
...
WebView browser =(WebView)findViewById(R.id.webby);
browser.getSettings().setJavaScriptEnabled(true);
...
```

Java Script Support

- The `WebView` widget is based on the Chromium web browser. Each and every Java Script library supported on the google chrome web browser will be supported on the `WebView`.

Java Script Support

- There are many different JavaScript libraries we can use in our hybrid application.

Java Script Support

- You can find samples for hybrid applications developed using SenchaTouch at http://dev.sencha.com/deploy/touch/examples/
- You can find samples for hybrid applications developed using jQueryMobile at http://www.jqmgallery.com

Java Script Support

- The following example displays a simple HTML document that uses the jQuery UI library.

Java Script Support

```
<html>

<head>
    <link href="http://ajax.googleapis.com/ajax/libs/
        jqueryui/1.8/themes/base/jquery-ui.css"
        rel="stylesheet"
        type="text/css"/>
    <script src=
        "http://ajax.googleapis.com/ajax/libs/jquery/1.4/jquery.min.js">
    </script>
    <script src=
        "http://ajax.googleapis.com/ajax/libs/jqueryui/1.8/jquery-
        ui.min.js">
    </script>
  <script>
  $(document).ready(function()
    {
            $("#tabs").tabs();
    });
  </script>
</head>
```

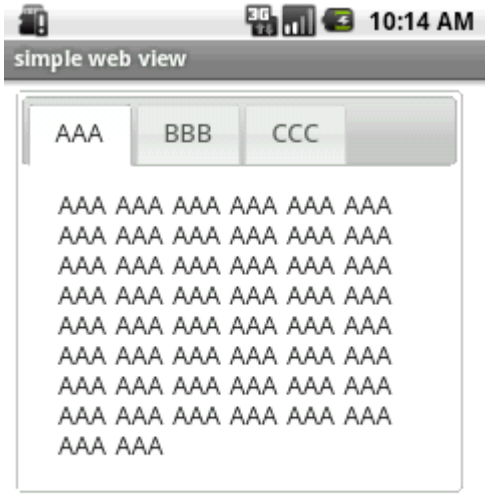
Java Script Support

```
<body>
<div id="tabs">
    <ul>
        <li><a href="#fragment-1"><span>AAA</span></a></li>
        <li><a href="#fragment-2"><span>BBB</span></a></li>
        <li><a href="#fragment-3"><span>CCC</span></a></li>
    </ul>
    <div id="fragment-1">
        AAA AAA AAA AAA AAA AAA
        AAA AAA AAA AAA AAA AAA
    </div>
    <div id="fragment-2">
        BBB BBB BBB BBB BBB BBB
        BBB BBB BBB BBB BBB BBB
    </div>
    <div id="fragment-3">
        CCC CCC CCC CCC CCC CCC
        CCC CCC CCC CCC CCC CCC
    </div>
</div>
</body>

</html>
```

Java Script Support

```
package com.abelski.samples;

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

public class WebViewSampleActivity extends Activity
{
    @Override
    public void onCreate(Bundle bndl)
    {
        super.onCreate(bndl);
        setContentView(R.layout.main);
        WebView browser =(WebView)findViewById(R.id.webby);
        browser.getSettings().setJavaScriptEnabled(true);
        Browser.
            loadUrl("http://www.abelski.com/courses/android/jq.html");
    }
}
```

Java Script Support



The `loadData()` Method

- Calling this method on our `WebView` object we can pass over a string that contains the data we want our web view object to parse and present as if it was retrieved over the web.

The `loadData()` Method

```
package com.abelski.samples;

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

public class WebViewSampleActivity extends Activity
{
    @Override
    public void onCreate(Bundle bndl)
    {
        super.onCreate(bndl);
        setContentView(R.layout.main);
        String str = "<body><h2>boga goga</h2><h4>gogo mogo";
        str += "lala</h4></body>";
        WebView browser =(WebView)findViewById(R.id.webby);
        browser.getSettings().setJavaScriptEnabled(true);
        browser.loadData(str,"text/html","UTF-8");
    }
}
```

The `loadData()` Method

The `WebView` Methods

- Calling `reload()` reloads the parsed data.
- Calling `goBack()` takes us back to the previous page in the browser history.
- Calling `goForward()` takes us forward one step in the browser history.
- Calling `canGoForward()` returns true if there is any history to to forward to.

The `WebView` Methods

- Calling `goBackOrForward()` goes back or forward in the browser history. Passing over a negative number causes going backward. Passing over a positive number causes going forward.
- Calling `canGoBackOrForward()` returns true if it is possible to go forward or backward the specified number of steps.

The `WebView` Methods

- Calling `clearHistory()` clears the browser history.
- Calling `clearCashe()` clears the browser cash memory.

The `WebViewClient` Class

- Each `WebView` object can be connected with a `WebViewClient` object.
- Calling the `setWebViewClient()` method on our `WebView` object passing over a reference for `WebViewClient` object we can put the two connected with each other. The supplied callback object will be notified of a wide range of activities.

The `WebViewClient` Class

- It is common to define a new class that extends `WebViewClient` and overrides the methods we are interested at.
- Overriding the `shouldOverrideUrlLoading()` method we can indirectly have our web view client handling various events that take place within the scope of the `WebView` object.

The `WebViewClient` Class

```
package com.abelski;

import java.util.*;
import android.os.*;
import android.app.*;
import android.webkit.*;

public class WebActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        String str = "";
        str += "<br><a href=\"clock\">system time</a>";
        str += "<br><a href=\"sdk\">sdk version</a>";
        str += "<br><a href=\"developer\">developer name</a>";
        WebView browser = (WebView) findViewById(R.id.webby);
        browser.getSettings().setJavaScriptEnabled(true);
        browser.setWebViewClient(new URLIntercepter());
        browser.loadData(str, "text/html", "UTF-8");
    }
```

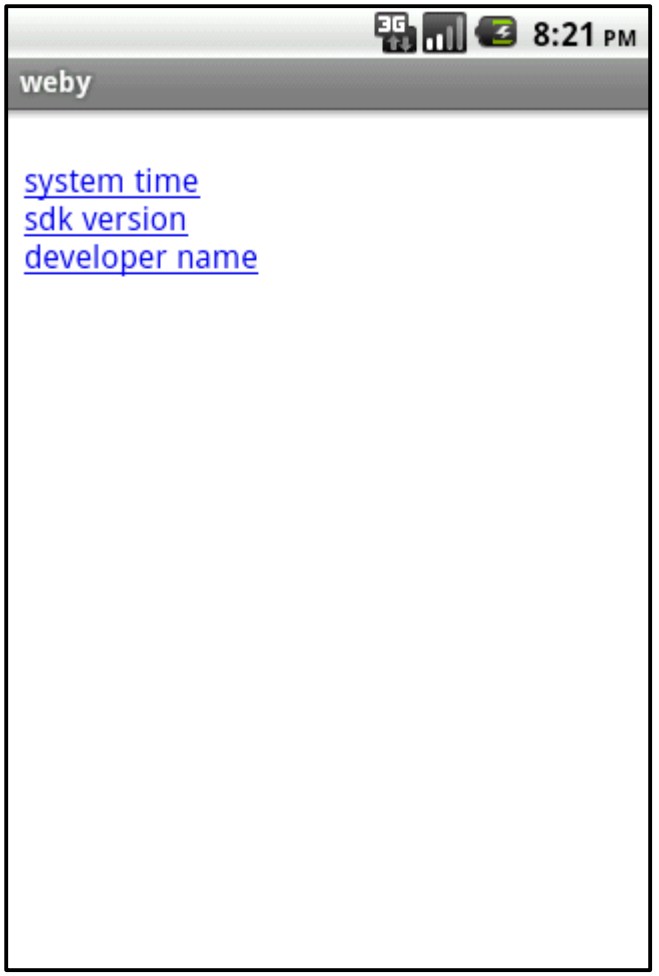
The `WebViewClient` Class

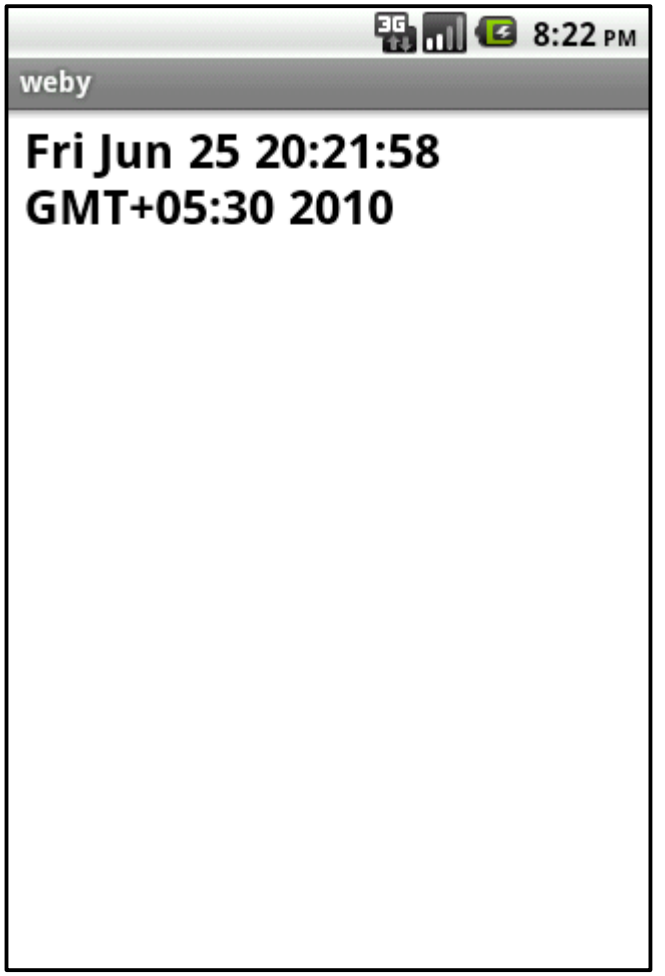
```
public class URLIntercepter extends WebViewClient
{
    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url)
    {
        if (url.contains("clock"))
        {
            String html = "<h2>" + new Date().toString() + "</h2>";
            view.loadData(html, "text/html", "UTF-8");
            return true;
        }
        else if(url.contains("sdk"))
        {
            String html = "<h2>The SDK version is " +
                Build.VERSION.SDK_INT + "</h2>";
            view.loadData(html, "text/html", "UTF-8");
            return true;
        }
```

The `WebViewClient` Class

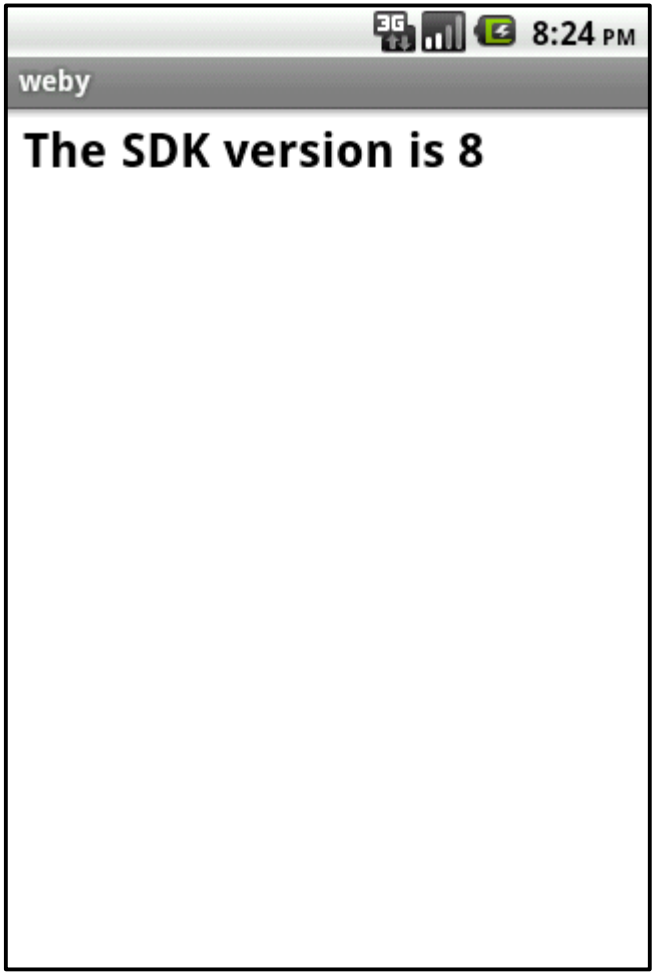
```
            else if(url.contains("developer"))
            {
                String html = "<h2>Developer name is Haim Michael</h2>";
                view.loadData(html, "text/html", "UTF-8");
                return true;
            }
            else
            {
                return false;
            }
        }
    }
}
```

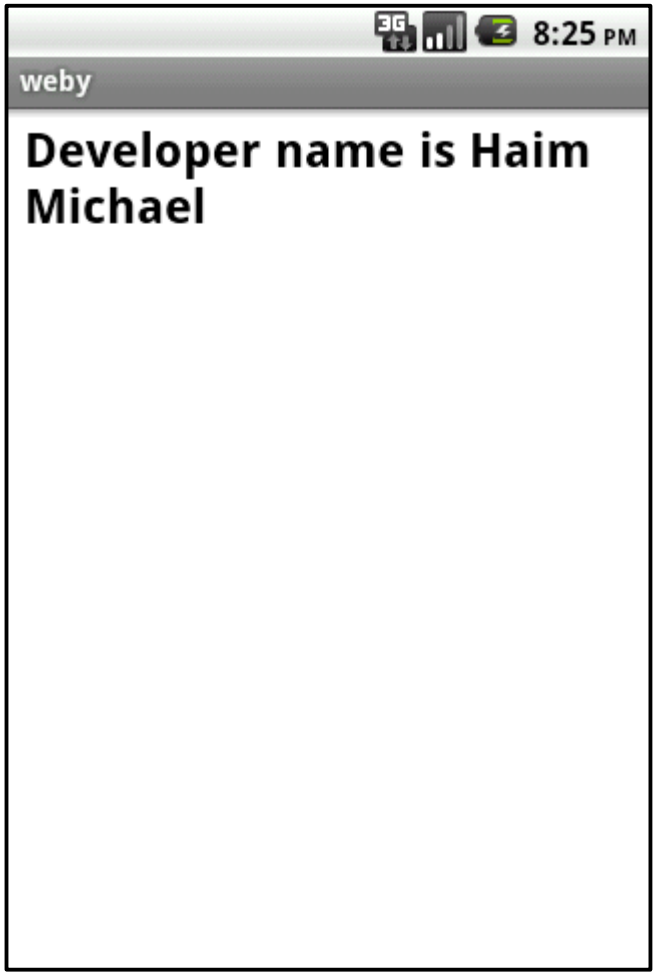
The `WebViewClient` Class





The `WebViewClient` Class





The `WebChromeClient` Class

- Similarly to `WebViewClient`, each `WebView` object can be connected with a `WebChromeClient` object.
- Calling the `setWebChromeClient()` method on our `WebView` object passing over a reference for `WebChromeClient` object we can put the two connected. The supplied callback object will be notified for a wide range of activities.

The `WebChromeClient` Class

- It is common to define a new class that extends `WebChromeClient` and overrides the methods we are interested at.

The `addJavascriptInterface()` Function

- Calling this method we can bind an object to the JavaScript execution code allowing code in JavaScript to call methods on that object.

```
public void addJavascriptInterface(
            Object obj, String interfaceName)
```

The name to use within the Java Script code

Tha Java class instance we want to expose

The `addJavascriptInterface()` Function

- In addition, each and every method defined in Java we want to allow its invocation from code written in JavaScript must be marked with the `@android.webkit.JavascriptInterface` annotation.

```
class CalculateObject
{
    @android.webkit.JavascriptInterface
        public int calculateSum(int numA, int
numB)
    {
        return numA + numB;
    }
}
```

The `addJavascriptInterface()` Function

```
public class HybridActivity extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        CalculateObject calcObject = new CalculateObject();
        super.onCreate(savedInstanceState);
        WebView webView = new WebView(this);
        webView.loadUrl("http://www.abelski.com/courses/android/simple.html");
        webView.getSettings().setJavaScriptEnabled(true);
        webView.addJavascriptInterface(calcObject, "ob");
        setContentView(webView);
    }
    class CalculateObject
    {
        @android.webkit.JavascriptInterface
        public int calculateSum(int numA, int numB)
        {
            return numA + numB;
        }
    }
}
```

The `addJavascriptInterface()` Function

```
<html>
    <head>
        <script>
        function calc()
        {
            var a = parseInt(document.myform.num_a.value,10);
            var b = parseInt(document.myform.num_b.value,10);
            var sum = window.ob.calculateSum(a,b);
            document.myform.result.value = sum;
        }
        </script>
    </head>
    <body>
        <form name="myform">
            <br/>number 1: <input type="text" name="num_a"/>
            <br/>number 2: <input type="text" name="num_b"/>
            <br/><input type="button" onclick="calc()" value="+"/>
            <br/>result: <input type="text" name="result"/>
        </form>
    </body>
</html>
```

The `addJavascriptInterface()` Function

Activity Life Cycle

- When the activity is paused we would like to pause the execution of the Java Script code in our WebView.

Activity Life Cycle

- We can achieve it adding into the `onResume()` and the `onPause()` call back functions the following code.

```
@Override
public void onPause()
{
    super.onPause();
    web.getSettings().setJavaScriptEnabled(false);
}

@Override
public void onResume()
{
    super.onResume();
    web.getSettings().setJavaScriptEnabled(true);
}
```

Activity Life Cycle

HybridActivity


```
package com.abelski.samples;

public class HybridActivity extends Activity
{
    private WebView web;

    @Override
    public void onPause()
    {
        super.onPause();
        web.getSettings().setJavaScriptEnabled(false);
    }

    @Override
    public void onResume()
    {
        super.onResume();
        web.getSettings().setJavaScriptEnabled(true);
    }

}
```

Activity Life Cycle

```
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_hybrid);
    web = (WebView) findViewById(R.id.webView1);
    //web.getSettings().setJavaScriptEnabled(true);
    web.loadUrl("file:///android_asset/www/index.html");
    web.addJavascriptInterface(new Logic(), "ob");
    Button bt = (Button) findViewById(R.id.button1);
    bt.setOnClickListener(new View.OnClickListener()
    {
        @Override
        public void onClick(View v)
        {
            Intent intent = new Intent(HybridActivity.this,
                    com.abelski.samples.AnotherActivity.class);
            startActivity(intent);
        }
    });

}
```

Activity Life Cycle

```
public class Logic
{
    public void writeToLog(String tag,String str)
    {
        Log.i(tag,str);
    }
}
```

Activity Life Cycle

AnotherActivity

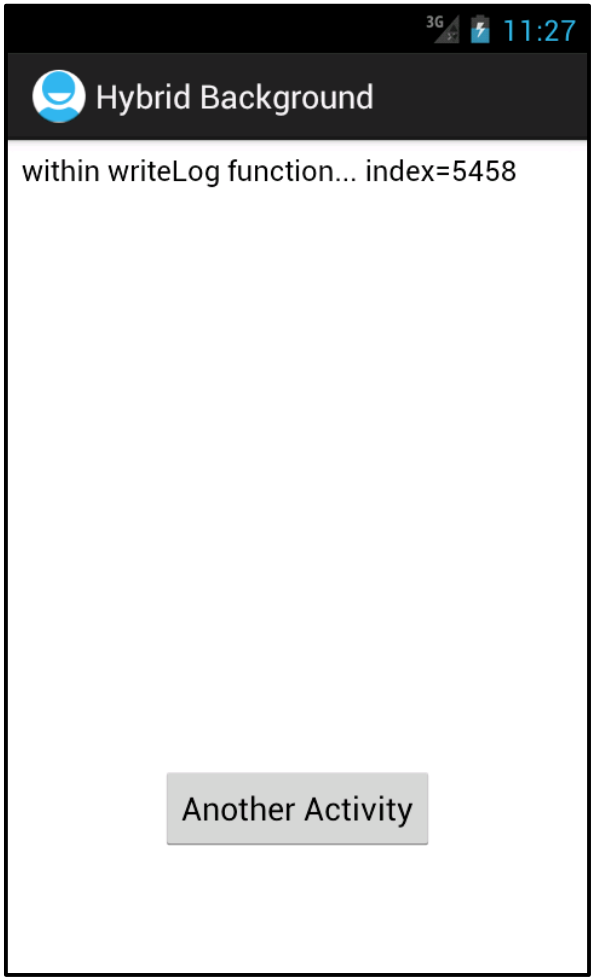
```
package com.abelski.samples;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class AnotherActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        TextView text = new TextView(this);
        text.setText("another activity");
        text.setTextSize(20);
        this.setContentView(text);
    }
}
```

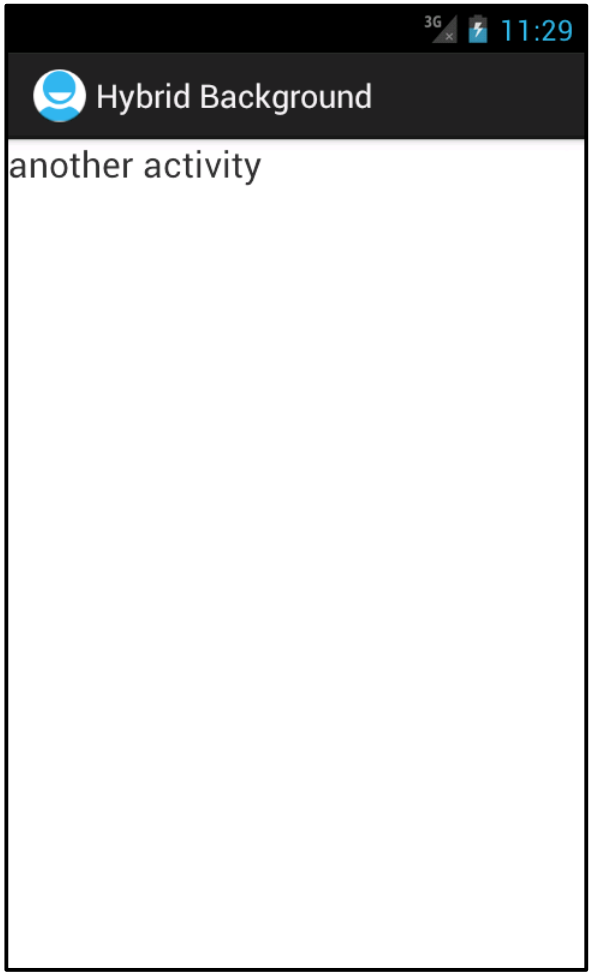
Activity Life Cycle

index.html

```
<script type="text/javascript">
index = 1;
setInterval(function(){writeLog();},1000);
function writeLog()
{
    var msg = "within writeLog function... index="+index;
    window.ob.writeToLog("js",msg);
    document.getElementById("msg").innerHTML = msg;
    index++;
}
</script>
<div id="msg">...</div>
```

Activity Life Cycle





Calling Functions in JavaScript

- We can use the loadUrl (or loadData) methods for calling functions that were defined in JavaScript.

```
webView.loadUrl("javascript:increment()");
```

Calling Functions in JavaScript

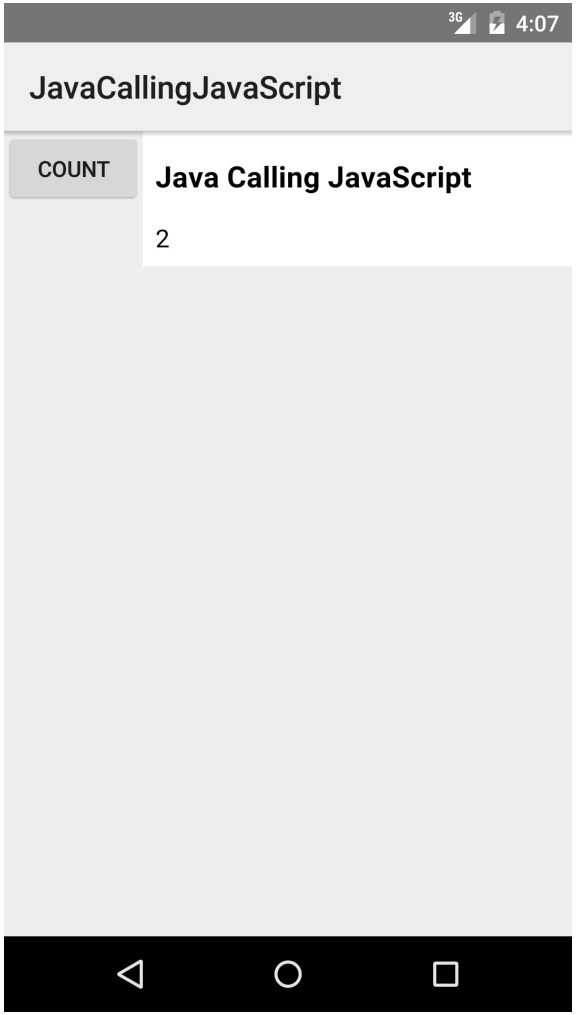
```
public class JavaCallingJavaScript extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        LinearLayout layout = new LinearLayout(this);
        final WebView webView = new WebView(this);
        webView.getSettings().setJavaScriptEnabled(true);
        webView.loadUrl("file:///android_asset/demo3.html");
        Button bt = new Button(this);
        bt.setText("count");
        bt.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                webView.loadUrl("javascript:increment()");
            }
        });
        layout.addView(bt);
        layout.addView(webView);
        setContentView(layout);
    }
}
```

Calling Functions in JavaScript

demo3.html

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta charset="UTF-8">
    <title></title>
</head>
<body>
<h3>Java Calling JavaScript</h3>
<div id="msg">0</div>
<script>
    function increment()
    {
        var ob = document.getElementById("msg");
        ob.innerText = parseInt(ob.innerText)+1;
    }
</script>
</body>
</html>
```

Calling Functions in JavaScript



Calling Functions in JavaScript

- We can alternatively invoke the `evaluateJavascript` method on the `WebView` object we use.
- The first argument is the code in JavaScript we want to execute (e.g. code that calls a function in JavaScript).
- The second argument is a reference for a listener object on which the `onReceiveValue` will be invoked when the code in JavaScript completes and returns a value.

Calling Functions in JavaScript

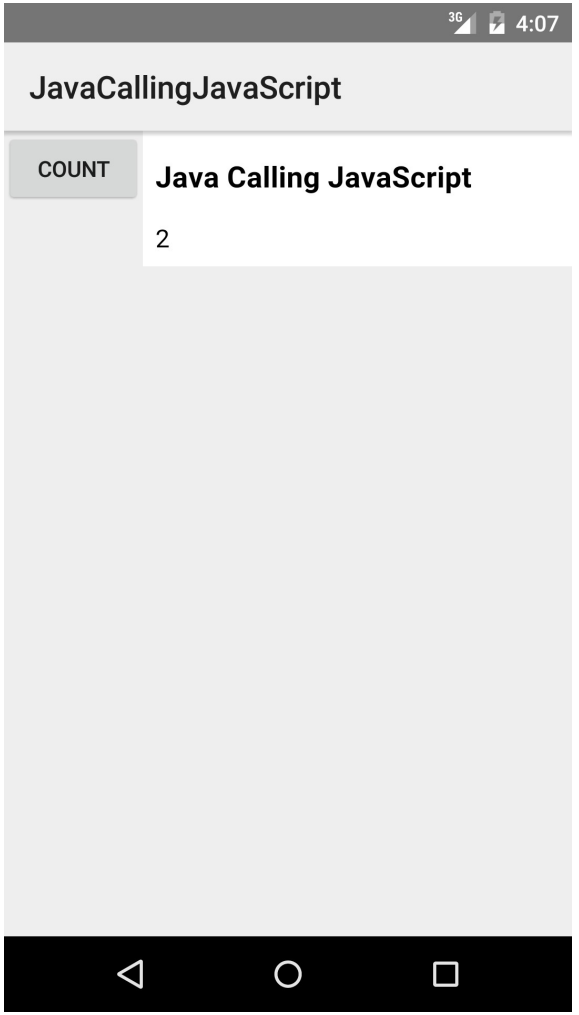
```
public class JavaCallingJavaScript extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        LinearLayout layout = new LinearLayout(this);
        final WebView webView = new WebView(this);
        webView.getSettings().setJavaScriptEnabled(true);
        webView.loadUrl("file:///android_asset/demo3.html");
        Button bt = new Button(this);
        bt.setText("count");
        bt.setOnClickListener(new View.OnClickListener()
        {
            webView.evaluateJavascript("increment()",
                new ValueCallback<String>()
                {
                    @Override
                    public void onReceiveValue(String value)
                    Log.i("rcv",value);
                 }
            });
        });
        layout.addView(bt);
        layout.addView(webView);
        set`ContentView(layout);
    }
}
```

Calling Functions in JavaScript

demo3.html

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta charset="UTF-8">
    <title></title>
</head>
<body>
<h3>Java Calling JavaScript</h3>
<div id="msg">0</div>
<script>
    function increment()
    {
        var ob = document.getElementById("msg");
        ob.innerText = parseInt(ob.innerText)+1;
    }
</script>
</body>
</html>
```

Calling Functions in JavaScript



Debugging Code in JavaScript

- We can debug the code in JavaScript running inside the web view using the google chrome dev tools.
- In order to enable it, we should first invoke the `setWebContentsDebuggingEnabled` static method that was defined in `WebView`.

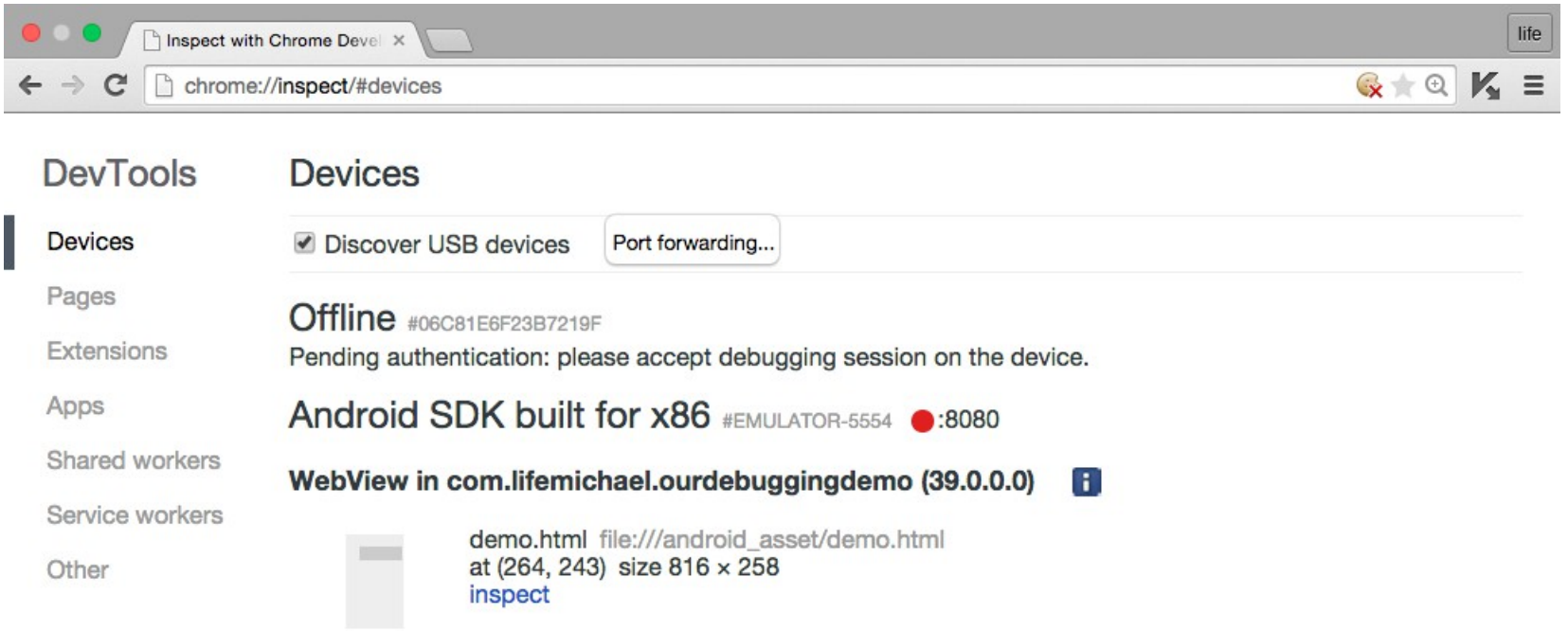
```
WebView.setWebContentsDebuggingEnabled(true);
```

Debugging Code in JavaScript

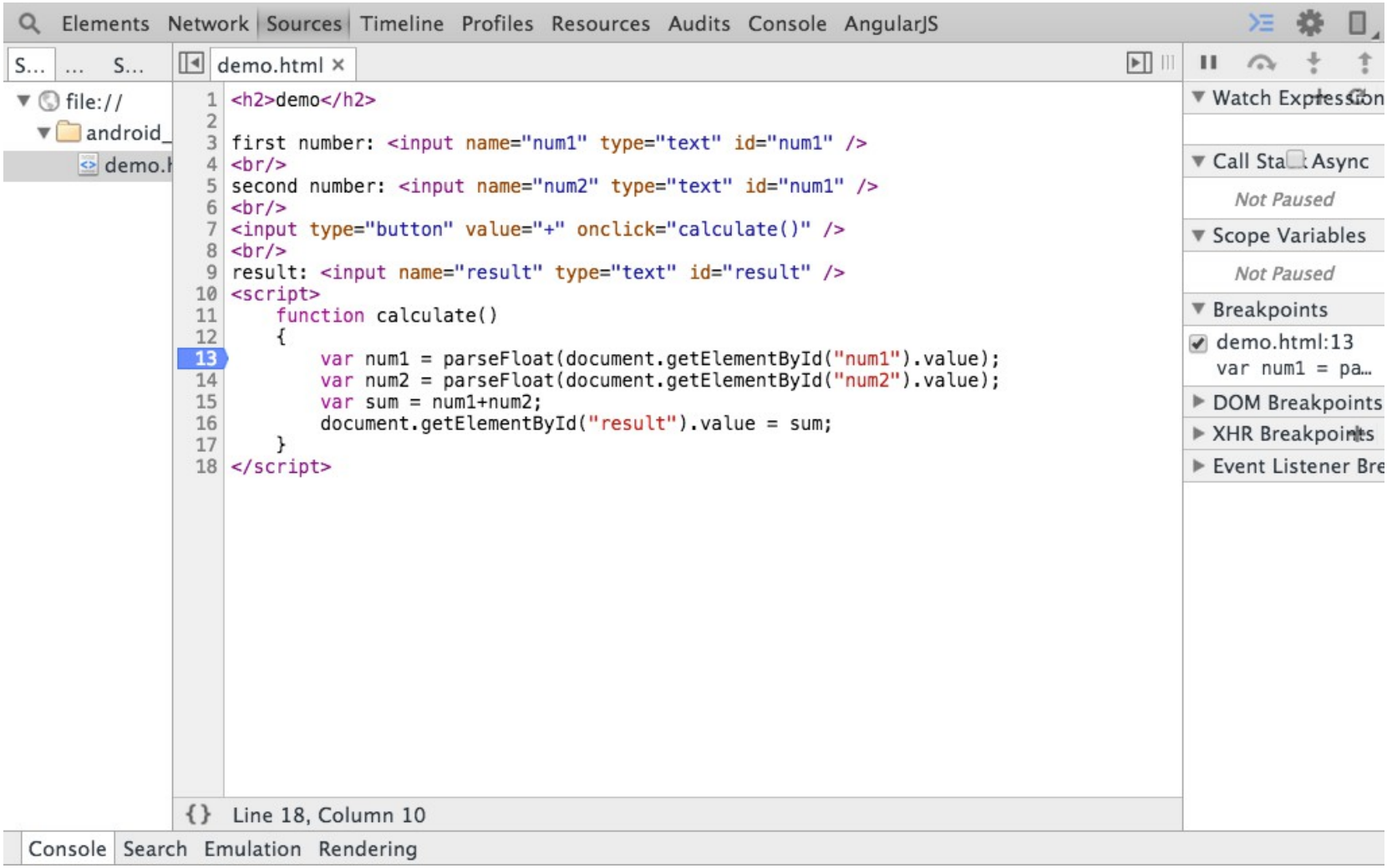
- We should open Chrome web browser and browse at the following URL address:

  `chrome://inspect/#devices`

Debugging Code in JavaScript



Debugging Code in JavaScript



Handling The Back Button

- When the user presses the device's back button he is taken to the previous activity.
- We can override this normal behavior by overriding the `onBackPresses()` function, that was defined in `Activity`.

```
…
public onBackPresses() {
    webView.loadUrl(…);
}
```