Introduction

The Android platform provides tools for developing web based applications. One of these tools is the WebView widget in the Android.webkit package. The WebView widget displays web pages and allows the user to interact with web content. This article describes the features of WebView widget.

Permission

Internet permission is required for the WebView widget to access the internet:

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

[Code 1]

Loading URL

WebView's primary function is to load a web page. This is done through the use of a loadUrl(String url) method:

```
loadUrl(String url) – Loads web page indicated by url parameter.
```

[Code 2]

Example

```
WebView myWebView = (WebView) findViewById(R.id.WebView);
myWebView.loadUrl("http://developer.samsung.com/resources.do");
WebView.loadUrl("file:///Android_res/raw/test.HTML");//from raw folder
WebView.loadUrl("file:///Android_asset/test.HTML");//from asset folder
```

[Code 3]

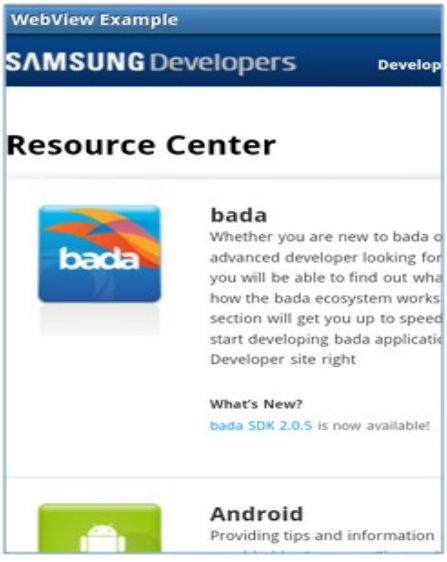


Figure 1: Loading a Web page

Loading Raw HTML

WebView also loads raw markup data using loadData (String data, String mimeType, String encoding)

Example

```
String rawHTML = "<HTML>"+
                        "<body><h1>Hello Android </h1></body>"+
                        "</HTML>";
WebView.loadData(data, "text/HTML", "UTF-8");
```

[Code 4]

FFigure 2: Rendering Raw HTML

Controls in WebView

WebSettings

WebView loads web pages and renders raw HTML data, but it is not a full-featured browser. As shown in the first two figures, it does not display controls like Title and navigation controls.

If a user clicks on a link in WebView, it by default opens up in the default Android browser, not in Web View. WebView by default disables various controls like zooming, JavaScript, modification of user agent string, and so forth. However, it allows the developer to do update or modify the default controls through WebSettings. The developer can obtain the WebSettings object through the getSettings() method.

```
WebSettings webSettings = WebView.getSettings();
//if web page uses JavaScript enabled it to load
webSettings.setJavaScriptEnabled (boolean enable);
webSettings.setUserAgentString(String ua);
```

[Code 5]

Similarly, there are other web methods for enabling or updating default value settings.

WebViewClient and WebChromeClient

These are two other client controls that can be used in WebView:

WebViewClient allows listening for web page events. Events such as when the web page begins to load, page has finished loading, when there is an error related to page loading, on form submission, links clicked by the end user, and other events.
WebChromeClient allows listening to JavaScript calls, notification of the current page such as console messages, alerts, progress updates of page, and other JavaScript calls.

Handling WebViewClient Events

The following example shows how the WebView and TextView widgets are used. WebView loads the web page and TextView sets the Title of the web page. Here it does the following:

1. Displays the address of the web page
2. Controls navigation

The corresponding xml file is:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
    Android:layout_width="match_parent"
    Android:layout_height="match_parent"
    Android:orientation="vertical" >
    <TextView
        Android:id="@+id/titlebar"
        Android:layout_width="match_parent"
        Android:layout_height="wrap_content"
        Android:textColor="#ffff00"
        Android:textSize="16sp"
        Android:text="Web Page Loading... " />
    <WebView
        Android:id="@+id/myWebView"
        Android:layout_width="match_parent"
        Android:layout_height="match_parent" />
</LinearLayout>
Extend WebViewClient class and override the corresponding methods:
public class WebViewActivity extends Activity {
        private WebView browser;
        private TextView titlebar;
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.WebViewclient);
                titlebar = (TextView)findViewById(R.id.titlebar);
                browser = (WebView) findViewById(R.id.myWebView);
```

```
//enable JavaScript
browser.getSettings().setJavaScriptEnabled(true);
//enable zoom controls
                browser.getSettings().setBuiltInZoomControls(true);
                browser.setWebViewClient(new MyWebViewClient());
                browser.loadUrl("http://developer.samsung.com/resources.do");
        }
        @Override
        public boolean onKeyDown(int keyCode, KeyEvent event) {
                if ((keyCode == KeyEvent.KEYCODE_BACK) &&
browser.canGoBack()) {//check for browser web page history
                        browser.goBack(); // back history item
                        return true;
                }
                return super.onKeyDown(keyCode, event);
        }
        private class MyWebViewClient extends WebViewClient {
                @Override
        public boolean shouldOverrideUrlLoading(WebView view, String url) {
                        titlebar.setText("Web Page Loading...");
                        browser.loadUrl(url);
                        return true;
                }
                @Override
                public void onPageFinished(WebView view, String url) {
                        // TODO Auto-generated method stub
                        super.onPageFinished(view, url);
                        titlebar.setText(url);
                }
                @Override
        public void onReceivedError(WebView view, int errorCode,
                String description, String failingUrl) {
        super.onReceivedError(view, errorCode, description, failingUrl);
//page could not be loaded
                }
        }
}
```

[Code 6]

shouldOverrideUrlLoading(WebView view, String url) ensures that the application gets control of the new web page one it finishes loading.

The onPageFinished(WebView view, String url) method is called once the web page loads. Here in this method the address of the web page is set to textview , once the web page finishes loading.

Similarly, canGoBack(),goBack(),canGoForward() are navigation methods which allow the user to move back and forth within the WebView page history.

Complete example is available at the end of the article.

Handling WebChromeClient Events

Consider an example consisting of the WebChromeClient and ProgressBar widgets. WebView loads the web page and its progress is displayed on activity progress bar.

```
public class WebChromeActivity extends Activity {
         private WebView browser;
         private TextView titlebar;
         /** Called when the activity is first created. */
         @Override
         public void onCreate(Bundle savedInstanceState) {
                  super.onCreate(savedInstanceState);
                  getWindow().requestFeature(Window.FEATURE_PROGRESS);
                  setContentView(R.layout.WebViewclient);
                  titlebar = (TextView)findViewById(R.id.titlebar);
                  titlebar.setVisibility(View.GONE);
                  browser = (WebView) findViewById(R.id.myWebView);
                  browser.getSettings().setJavaScriptEnabled(true);
                  browser.getSettings().setBuiltInZoomControls(true);
//set webchrome client
                  browser.setWebChromeClient(new MyWebChromeClient());
                  browser.loadUrl("http://developer.samsung.com/resources.do");

         }

         private class MyWebChromeClient extends WebChromeClient {
         @Override
         public void onProgressChanged(WebView view, int newProgress) {
                           // TODO Auto-generated method stub
                           super.onProgressChanged(view, newProgress);
                           setProgress(newProgress * 100);
                  }
         }
}
```

[Code 7]

The onProgressChanged(WebView view, int newProgress) method is called as the web page loads in WebView. The value is set to setProgress() method of activity. The output is shown in Figure 3.

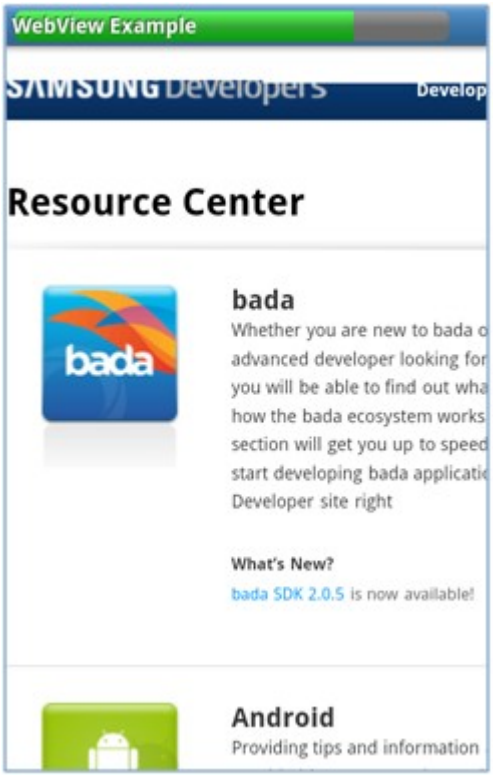


Figure 3: WebChromeClient Example

Handling JavaScript Events

The following example shows the handling of JavaScript events in the WebChromeClient class. A sample HTML file is created and stored in asset folder of the project. Here are the details of the HTML file:

```
<HTML>
<head>
<script type="text/JavaScript">
function showAlert()
{
alert("This is js alert method");
}

function showPrompt() {
prompt("This is js prompt method", "")
}
function showConfirm() {
confirm("This is js confirm method")
}
</script>
</head>
<body>
```

```
<button onclick="showAlert()">Alert Button</button><br>
<button onclick="showConfirm()">Confirm Button</button><br>
<button onclick="showPrompt()">Prompt Button</button><br>
</center>
</body>
</HTML>
```

[Code 8]

It consists of three buttons - Alert, Confirm, and Prompt buttons as shown in Figure 4. The corresponding JavaScript methods are implemented in the script tag. The HTML file is loaded in WebView using the loadUrl("file:///Android_asset/testJavaScript.HTML") method.

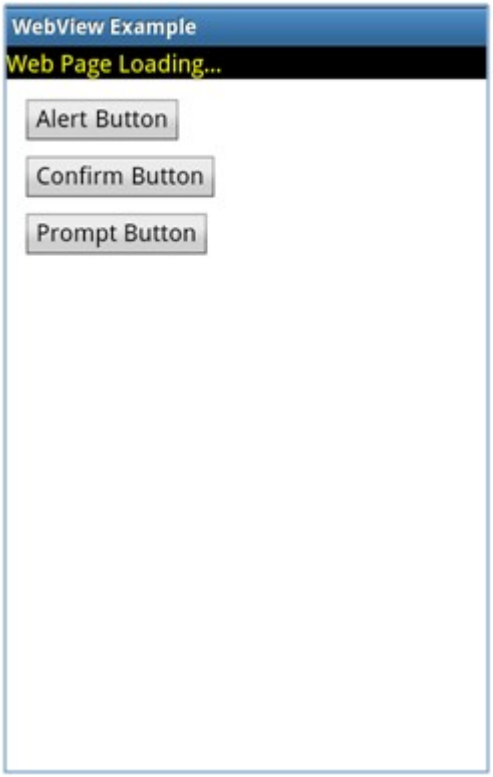


Figure 4: WebView JavaScript Example

If the user clicks on any of these buttons, then the corresponding WebChromeClient methods are called. For example when clicking the Alert Button, the showAlert() method is called. showAlert() implements the alert() function which in turn is called by the WebChromeClient class onJsAlert(WebView view, String url, String message, JsResult result). The parameter in the alert() function is passed to the message parameter of the onJSAlert() method.

```
public class WebViewJavaScriptActivity1 extends Activity {
        private WebView browser;
        private TextView titlebar;
        @Override
        public void onCreate(Bundle savedInstanceState) {
```

```
                super.onCreate(savedInstanceState);
                setContentView(R.layout.WebViewclient);
                titlebar = (TextView)findViewById(R.id.titlebar);
                browser = (WebView) findViewById(R.id.myWebView);
                browser.getSettings().setJavaScriptEnabled(true);
                browser.setWebChromeClient(new MyWebChromeClient());
//load HTML file from asset folder
                browser.loadUrl("file:///Android_asset/testJavaScript.HTML");
        }
private class MyWebChromeClient extends WebChromeClient {
        @Override
        public boolean onJsAlert(WebView view, String url, String message,
                                final JsResult result) {
                new AlertDialog.Builder(WebViewJavaScriptActivity1.this)
                        .setTitle("JavaScript Alert !")
                        .setMessage(message)
                        .setPositiveButton(Android.R.string.ok,
                            new AlertDialog.OnClickListener() {
                public void onClick(DialogInterface dialog, int which) {
                                    // do your stuff here
                                    result.confirm();
                                }
                            }).setCancelable(false).create().show();
                         return true;
                }
        @Override
        public boolean onJsConfirm(WebView view, String url, String message,
                                final JsResult result) {
                new AlertDialog.Builder(WebViewJavaScriptActivity1.this)
                        .setTitle("JavaScript Confirm Alert !")
                        .setMessage(message)
                        .setPositiveButton(Android.R.string.ok,
                            new AlertDialog.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                                    // do your stuff here
                                    result.confirm();
                                }
                            }).setCancelable(false).create().show();
                         return true;
                }
        @Override
        public boolean onJsPrompt(WebView view, String url, String message,
                         String defaultValue, final JsPromptResult result) {
```

```
                new AlertDialog.Builder(WebViewJavaScriptActivity1.this)
                        .setTitle("JavaScript Prompt Alert !")
                        .setMessage(message)
                        .setPositiveButton(Android.R.string.ok,
                            new AlertDialog.OnClickListener() {
                        public void onClick(DialogInterface dialog, int which) {
                                    // do your stuff here
                                    result.confirm();
                                }
                            }).setCancelable(false).create().show();
                         return true;
                }
        }
}
```

[Code 9]

The output of each button click is shown in Figure 5.

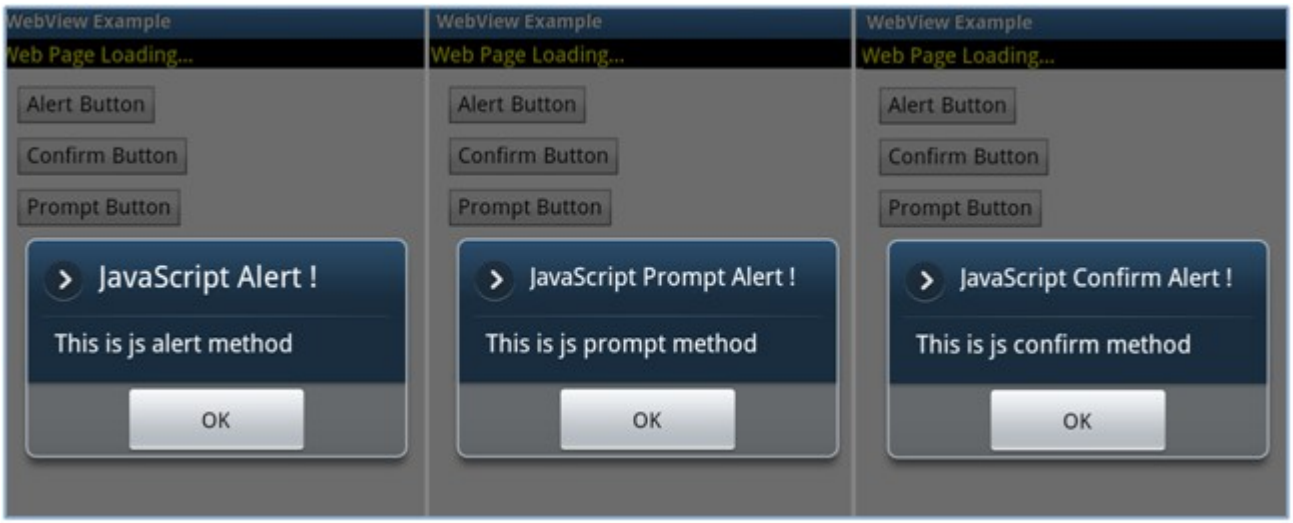


Figure 5: JavaScript Example Output

JavaScript-Android Binding

WebView allows JavaScript code to bind to Android code through an interface. The developer needs to create the interface and bind it using addJavaScriptInterface (Object obj, String interfaceName) method. It takes a class instance to bind to JavaScript and the interface name for JavaScript to access the class.

> JavaScript to Android binding is a very useful feature but can be risky at times. You should use addJavaScriptInterface(...) only if the web page is written yourself. If the web page is not trustworthy, then an attacker can inject HTML commands and execute the code, leading to security concerns.

Here is an example demonstrating the JavaScript-Android binding. The following example consists of HTML page with JavaScript code stored in an asset folder and Android activity. The output of the

example is shown in Figure 6.

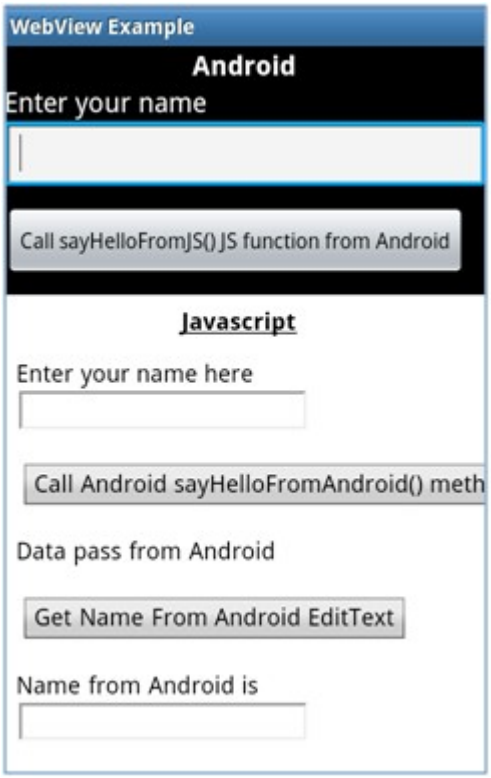


Figure 6: JavaScript-Android Binding Example

It consists of an Activity with the following:

- •An EditText field asking the user to enter their name
- •A button Call sayHelloFromJS() JS function from Android to call the JavaScript method from the activity class.
- •WebView to render the HTML page.

The HTML page used in WebView consists of:

- •A TextField asking user to enter name
- •A button labeled Call Android sayHelloFromAndroid method from JS to call the Android method from JavaScript.
- •A button labeled Get Name from Android EditText to call the Android method to retrieve the name entered in Android EditText widget.
- •A TextField to display the retrieved name from Android the EditText widget

```
<HTML>
<center><b><u>JavaScript</u></b></center>
<script language="JavaScript">
function getData() {
document.getElementById("namefromAndroid").value =
```

```
locater.getNameFromAndroidET();
var HTML = ""+document.getElementById("namefromAndroid").value;
alert("JavaScript says: Name entered in Android EditText is "+HTML);
}
function setData() {
var HTML = ""+document.getElementById("namefromjs").value;
locater.sayHelloFromAndroid(HTML);
}
function sayHelloFromJS(value) {
alert("JavaScript says: Hello "+value+" !!! How are you?");
}
</script>
</head>
<body>
<p> Enter your name here <input type="text" id="namefromjs" />
<p> <input type="button" onclick= "setData()" value="Call Android
sayHelloFromAndroid() method from JS">
<p> Data pass from Android</p>
<p> <input type="button" onclick= "getData()" value="Get Name From Android
EditText">
<p> Name from Android is <input type="text" id="namefromAndroid" />
</body>
</HTML>
```

[Code 10]

As shown in the above HTML code, The button Call Android sayHelloFromAndroid() method from JS calls setData(). setData() reads the name from the textfield and calls an Android method using the locater.sayHelloFromAndroid(HTML) method.

Similarly, the Get Name from Android button calls the getData() method, which calls the locater.getNameFromAndroidET() Android method and sets the name to its namefromAndroid variable.

The Android method is called using the locater.MethodName method where locater is the name of the interface defined in Android Activity for binding.

WebView uses addJavaScriptInterface(JSAndroidBindingClass, "locater") and intakes the JSAndroidBindingClass class and the interface named locater for accessing the Android class. This creates an interface called locater for JavaScript running in the WebView. At this point, web application has access to the JSAndroidBindingClass class.

The code snippet of WebViewJavaScriptActivity activity class follows.

```
public class WebViewJavaScriptActivity extends Activity {
private JSTest JSAndroidBindingClass = new JSTest();
private WebView browser;
```

```
private EditText nameET;
private Button jsButton;

@Override
public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.main);
jsButton = (Button) findViewById(R.id.button1);
nameET = (EditText) findViewById(R.id.editText1);
         browser = (WebView) findViewById(R.id.WebView1);
browser.addJavaScriptInterface(JSAndroidBindingClass, "locater");
         browser.loadUrl("file:///Android_asset/test.HTML");
jsButton.setOnClickListener(new OnClickListener() {
                          @Override
                          public void onClick(View v) {
                                   // TODO Auto-generated method stub
                                   AndroidETName = nameET.getText().toString();
                                   callJSHelloMethod(AndroidETName);
                          }
                 });
}
private void callJSHelloMethod(String name) {
         browser.loadUrl("JavaScript:sayHelloFromJS('" + name + "')");
}
public class JSTest {
         public String getNameFromAndroidET() {
                  return AndroidETName;
         }
         public void sayHelloFromAndroid(String actualData) {
                  new AlertDialog.Builder(WebViewJavaScriptActivity.this)
                  .setTitle("Android method called from JavaScript !")
                  .setMessage("Android Says: Hello " + actualData
                                                      + " !!! How are you.")
                  .setPositiveButton(Android.R.string.ok,
new AlertDialog.OnClickListener() {
                                   public void onClick(DialogInterface dialog,
                                                                        int which) {
                                            // do your stuff
                          }                                                    }).setCan
celable(false).create().show();
                  }
         }
}
```

[Code 11]

Android to JavaScript Function Calls

JavaScript methods from Android are called using JavaScript:Method Name where Method Name is the name of JavaScript method. All JavaScript methods are called using WebView loadUrl() method. As shown in the activity class, the Android Button Call sayHelloFromJS() JS function from Android, callJSHelloMethod () gets called which in turn calls the JavaScript sayHelloFromJS () method using browser.loadUrl("JavaScript:sayHelloFromJS('" + name + "')");
The output is shown in Figure 7.

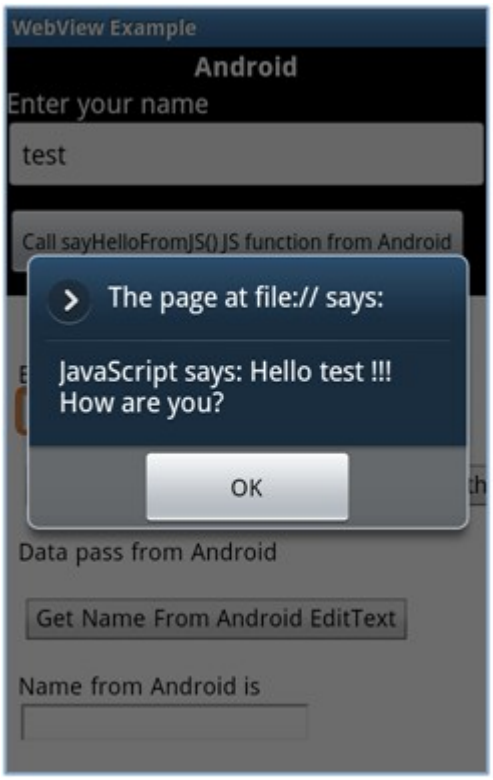


Figure 7: Android to JavaScript Function Call

JavaScript to Android function calls

As shown in sample Code 11, JavaScript calls the JSTest class that contains methods using locater as interface name. Those are locater.sayHelloFromAndroid(HTML) and locater.getNameFromAndroidET() for HTML buttons Call Android sayHelloFromAndroid() method from JS and Get Name From Android EditText.
The output is shown in Figure 8 with name entered as “tom” in the JavaScript text field followed by the Call Android sayHelloFromAndroid() method from JS button.

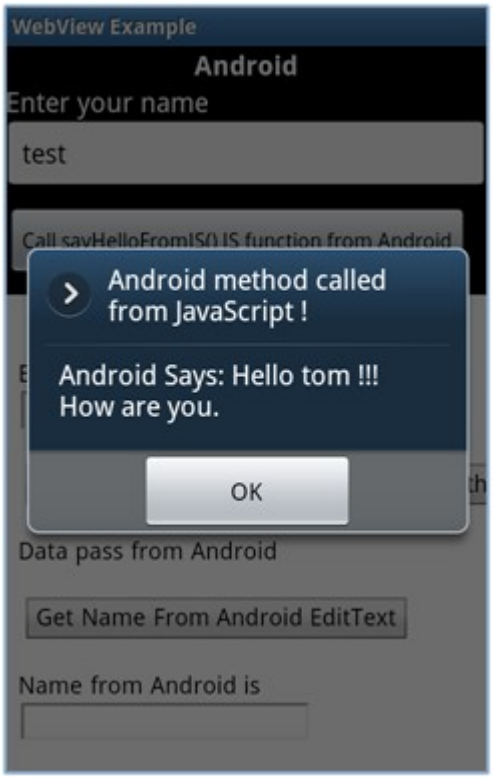


Figure 8: JavaScript to Android Method Call

Example

The example shows four different activities. Each Activity demonstrates a feature of WebView.

WebViewActivity.java

```
package test.WebView.JavaScript;
import Android.app.Activity;
import Android.os.Bundle;
import Android.view.KeyEvent;
import Android.webkit.WebView;
import Android.webkit.WebViewClient;
import Android.widget.TextView;
public class WebViewActivity extends Activity {

        private WebView browser;
        private TextView titlebar;
        /** Called when the activity is first created. */
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.WebViewclient);
                titlebar = (TextView) findViewById(R.id.titlebar);
                browser = (WebView) findViewById(R.id.myWebView);
```

```
                browser.getSettings().setJavaScriptEnabled(true);
                browser.getSettings().setBuiltInZoomControls(true);
                browser.setWebViewClient(new MyWebViewClient());
                browser.loadUrl("http://developer.samsung.com/resources.do");
        }
        @Override
        public boolean onKeyDown(int keyCode, KeyEvent event) {
        if ((keyCode == KeyEvent.KEYCODE_BACK) && browser.canGoBack()) {
                        browser.goBack();
                        return true;
                }
                return super.onKeyDown(keyCode, event);
        }
        private class MyWebViewClient extends WebViewClient {
                @Override
        public boolean shouldOverrideUrlLoading(WebView view, String url) {
                        titlebar.setText("Web Page Loading...");
                        browser.loadUrl(url);
                        return true;
                }
                @Override
                public void onPageFinished(WebView view, String url) {
                        // TODO Auto-generated method stub
                        super.onPageFinished(view, url);
                        titlebar.setText(url);
                }
                @Override
                public void onReceivedError(WebView view, int errorCode,
                                String description, String failingUrl) {
                        // TODO Auto-generated method stub
           super.onReceivedError(view, errorCode, description, failingUrl);
                }
        }
}
```

[Code 12]

WebChromeActivity.java

```
package test.WebView.JavaScript;
import Android.app.Activity;
import Android.os.Bundle;
import Android.view.View;
import Android.view.Window;
import Android.webkit.WebChromeClient;
```

```
import Android.webkit.WebView;
import Android.widget.TextView;
public class WebChromeActivity extends Activity {
        private WebView browser;
        private TextView titlebar;
        /** Called when the activity is first created. */
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                getWindow().requestFeature(Window.FEATURE_PROGRESS);
                setContentView(R.layout.WebViewclient);
                titlebar = (TextView)findViewById(R.id.titlebar);
                titlebar.setVisibility(View.GONE);
                browser = (WebView) findViewById(R.id.myWebView);
                browser.getSettings().setJavaScriptEnabled(true);
                browser.getSettings().setBuiltInZoomControls(true);
                browser.setWebChromeClient(new MyWebChromeClient());
                browser.loadUrl("http://developer.samsung.com/resources.do");
        }
        private class MyWebChromeClient extends WebChromeClient {
                @Override
        public void onProgressChanged(WebView view, int newProgress) {
                        // TODO Auto-generated method stub
                        super.onProgressChanged(view, newProgress);
                        setProgress(newProgress * 100);
                }
        }
}
```

[Code 13]

WebViewJavaScriptActivity1.java

```
package test.WebView.JavaScript;
import Android.app.Activity;
import Android.app.AlertDialog;
import Android.content.DialogInterface;
import Android.os.Bundle;
import Android.webkit.JsPromptResult;
import Android.webkit.JsResult;
import Android.webkit.WebChromeClient;
import Android.webkit.WebView;
import Android.widget.TextView;
public class WebViewJavaScriptActivity1 extends Activity {
        private WebView browser;
```

```
        private TextView titlebar;
        /** Called when the activity is first created. */
        @Override
        public void onCreate(Bundle savedInstanceState) {
                 super.onCreate(savedInstanceState);
                 setContentView(R.layout.WebViewclient);
                 titlebar = (TextView) findViewById(R.id.titlebar);
                 browser = (WebView) findViewById(R.id.myWebView);
                 browser.getSettings().setJavaScriptEnabled(true);
                 browser.getSettings().setBuiltInZoomControls(true);
                 browser.setWebChromeClient(new MyWebChromeClient());
                 browser.loadUrl("file:///Android_asset/testJavaScript.HTML");
        }
        private class MyWebChromeClient extends WebChromeClient {
        @Override
        public boolean onJsAlert(WebView view, String url, String message,
                                                              final JsResult result) {
                 new AlertDialog.Builder(WebViewJavaScriptActivity1.this)
                                              .setTitle("JavaScript Alert !")
                                              .setMessage(message)
                                              .setPositiveButton(Android.R.string.ok,
                          new AlertDialog.OnClickListener() {
                   public void onClick(DialogInterface dialog, int which) {
                                   // do your stuff
                                   result.confirm();
                          }
                          }).setCancelable(false).create().show();
                          return true;
                 }
                 @Override
        public boolean onJsConfirm(WebView view, String url, String message,
                                                              final JsResult result) {
                 new AlertDialog.Builder(WebViewJavaScriptActivity1.this)
                                              .setTitle("JavaScript Confirm Alert !")
                                              .setMessage(message)
                                              .setPositiveButton(Android.R.string.ok,
                 new AlertDialog.OnClickListener() {
                 public void onClick(DialogInterface dialog,     int which) {
                                   // do your stuff
                                   result.confirm();
                 }
}).setCancelable(false).create().show();
                 return true;
```

```
        }
        @Override
        public boolean onJsPrompt(WebView view, String url, String message,
                        String defaultValue, final JsPromptResult result) {
                new AlertDialog.Builder(WebViewJavaScriptActivity1.this)
                                        .setTitle("JavaScript Prompt Alert !")
                                        .setMessage(message)
                                        .setPositiveButton(Android.R.string.ok,
                        new AlertDialog.OnClickListener() {
                public void onClick(DialogInterface dialog,     int which) {
                                // do your stuff
                                result.confirm();
                        }

                        }).setCancelable(false).create().show();
                        return true;
                }
        }
}
```

[Code 14]

WebViewJavaScriptActivity.java

```
package test.WebView.JavaScript;
import Android.app.Activity;
import Android.app.AlertDialog;
import Android.content.DialogInterface;
import Android.os.Bundle;
import Android.view.MotionEvent;
import Android.view.View;
import Android.view.View.OnClickListener;
import Android.webkit.WebChromeClient;
import Android.webkit.WebView;
import Android.widget.Button;
import Android.widget.EditText;
public class WebViewJavaScriptActivity extends Activity {
        private WebView browser;
        private EditText nameET;
        private Button jsButton;
        private String AndroidETName;
        private JSTest JSAndroidBindingClass = new JSTest();
        /** Called when the activity is first created. */
        @Override
        public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        AndroidETName = "unknown";
        browser = (WebView) findViewById(R.id.WebView1);
        nameET = (EditText) findViewById(R.id.editText1);
        jsButton = (Button) findViewById(R.id.button1);
        browser.getSettings().setJavaScriptEnabled(true);
        browser.setWebChromeClient(new WebChromeClient());
browser.addJavaScriptInterface(JSAndroidBindingClass, "locater");
        browser.loadUrl("file:///Android_asset/test.HTML");
        browser.setOnTouchListener(new View.OnTouchListener() {
                @Override
                public boolean onTouch(View v, MotionEvent event) {
                        switch (event.getAction()) {
                        case MotionEvent.ACTION_DOWN:
                        case MotionEvent.ACTION_UP:
                                if (!v.hasFocus()) {
                                        v.requestFocus();
                                }
                                break;
                        }
                        return false;
                }
        });
        jsButton.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
                        // TODO Auto-generated method stub
                        AndroidETName = nameET.getText().toString();
                        if (AndroidETName == null)
                                AndroidETName = "unknown";
                        callJSHelloMethod(AndroidETName);
                        browser.requestFocus(View.FOCUS_DOWN);
                }
        });
}
private void callJSHelloMethod(String name) {
        browser.loadUrl("JavaScript:sayHelloFromJS('" + name + "')");
}
public class JSTest {
        public String getNameFromAndroidET() {
                return AndroidETName;
        }
```

```
                public void sayHelloFromAndroid(String actualData) {
                        if (actualData == null)
                                actualData = "unknown";
                        if (actualData.length() < 1)
                                actualData = "unknown";
                        new AlertDialog.Builder(WebViewJavaScriptActivity.this)
                        .setTitle("Android method called from JavaScript !")
                .setMessage("Android Says: Hello " + actualData + " !!! How are you.")
                        .setPositiveButton(Android.R.string.ok,
                        new AlertDialog.OnClickListener() {
                        public void onClick(DialogInterface dialog,int which) {
                                // do your stuff
                        }
}).setCancelable(false).create().show();
                }
        }
}
```

[Code 15]

Test.HTML (asset folder)

```
<HTML>
<center><b><u>JavaScript</u></b></center>
<script language="JavaScript">
function getData() {
document.getElementById("namefromAndroid").value = locater.getNameFromAndroidET();
var HTML = ""+document.getElementById("namefromAndroid").value;
alert("JavaScript says: Name entered in Android EditText is "+HTML);
}
function setData() {
var HTML = ""+document.getElementById("namefromjs").value;
locater.sayHelloFromAndroid(HTML);
}
function sayHelloFromJS(value) {
alert("JavaScript says: Hello "+value+" !!! How are you?");
}
</script>
</head>
<body>
<p> Enter your name here <input type="text" id="namefromjs" />
<p> <input type="button" onclick= "setData()" value="Call Android sayHelloFromAndroid()
method from JS">
<p> Data pass from Android</p>
<p> <input type="button" onclick= "getData()" value="Get Name From Android EditText">
```

```
<p> Name from Android is <input type="text" id="namefromAndroid" />
</body>
</HTML>
```

[Code 16]

Test.HTML (asset folder)

```
<HTML>
<head>
<script type="text/JavaScript">
function showAlert(){
 alert("This is js alert method"); }
function showPrompt() {
 prompt("This is js prompt method", "") }
function showConfirm() {
 confirm("This is js confirm method") }
</script>
</head>
<body>
<button onclick="showAlert()">Alert Button</button><br>
<button onclick="showConfirm()">Confirm Button</button><br>
<button onclick="showPrompt()">Prompt Button</button><br>
</center>
</body>
</HTML>
```

[Code 17]

main.xml (layout folder)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
    Android:layout_width="fill_parent"
    Android:layout_height="fill_parent"
    Android:orientation="vertical" >
    <TextView
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:text="Android"
        Android:textStyle="bold"
        Android:layout_gravity="center"
        Android:textAppearance="?Android:attr/textAppearanceMedium" />
    <TextView
        Android:id="@+id/textView1"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
```

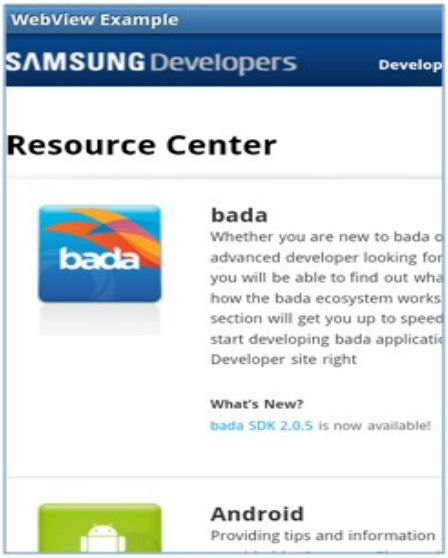
```
        Android:text="Enter your name"
        Android:textAppearance="?Android:attr/textAppearanceMedium" />
    <EditText
        Android:id="@+id/editText1"
        Android:layout_width="match_parent"
        Android:layout_height="wrap_content"
        Android:ems="10" >
        <requestFocus />
    </EditText>
    <Button
         Android:layout_marginTop="10dp"
        Android:id="@+id/button1"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:text="Call sayHelloFromJS() JS function from Android" />
    <WebView
        Android:layout_marginTop="10dp"
        Android:id="@+id/WebView1"
        Android:layout_width="match_parent"
        Android:layout_height="match_parent"
        Android:focusable="true"
        Android:focusableInTouchMode="true" />
</LinearLayout>
```

[Code 18]

WebViewclient.xml (layout folder)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
    Android:layout_width="match_parent"
    Android:layout_height="match_parent"
    Android:orientation="vertical" >
    <TextView
        Android:id="@+id/titlebar"
        Android:layout_width="match_parent"
        Android:layout_height="wrap_content"
        Android:text="Web Page Loading... "
        Android:textColor="#ffff00"
        Android:textSize="16sp" />
    <WebView
        Android:id="@+id/myWebView"
        Android:layout_width="match_parent"
        Android:layout_height="match_parent" />
</LinearLayout>
```

[Code 19]



ref:http://developer.samsung.com/android/technical-docs/WebView-in-Android