



AndroidとJavaScriptの調査

1.0版

システムセンス(株)

遠藤

目的

- JavaScriptで何ができる？
- 実際に動かすには？
- JavaScriptの危険性は？

AndroidはWebkitを搭載しているのでJavaScriptが利用できる

・Webkitとは？

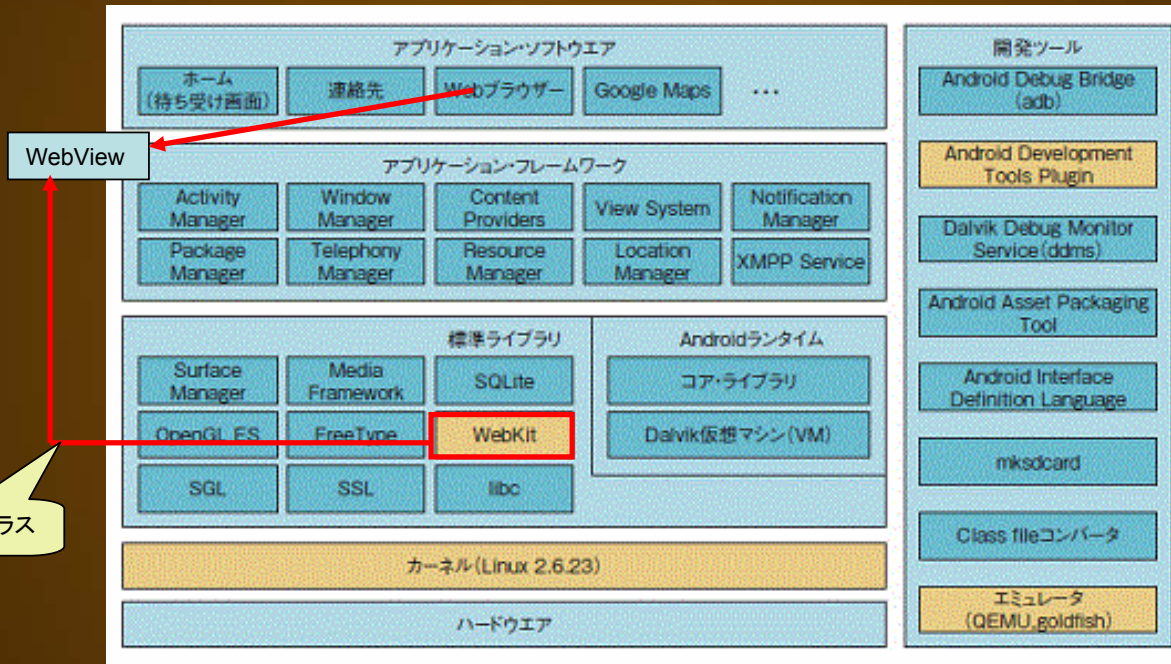
アップルが中心となって開発しているオープンソースのHTMLレンダリングエンジン群の総称。

JavaScriptの他、HTML,CSS,SVGなどを解釈できる。

Webページのレンダリングやアプリケーションとしてのインターフェイスを形成するためのフレームワークとして用いられている。

AndroidのブラウザもWebkitを利用している。

Androidアーキテクチャの構成図



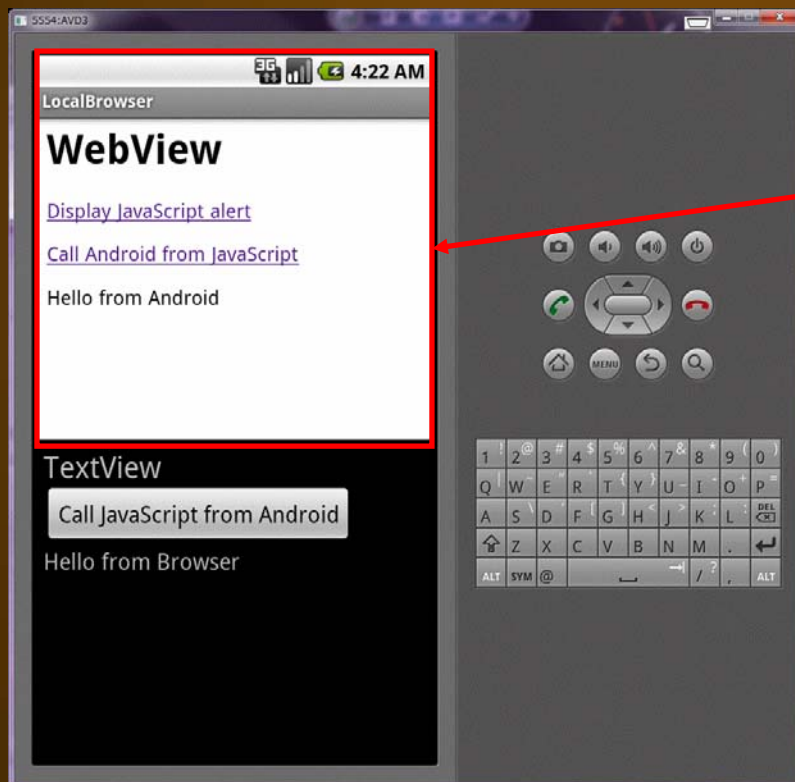
JavaScriptをAndroidアプリで使用するにはWebKitのWebViewというクラスを利用する。

WebViewクラス

- アンドロイドアプリからJavaScriptを利用するためのクラスでありブラウザ機能を提供してくれる。
- 1つのWebViewが1枚のHTML (WEBサイトの1ページ)に相当する。
- 自作のアプリケーションでHTMLを表示するにはWebViewを貼り付けることになる。(ブラウザ等)
- WebViewクラスを利用するにはコンストラクタを使ってWebViewクラスのオブジェクトを作成する。
- WebViewに「loadUrl」関数を使って表示するURLを指定する。

WebViewクラスのインスタンス1つにつき1枚のHTMLに対応するイメージ

(タブブラウザの場合タブ毎に1つ必要になる)



WebView

```
webView.loadUrl("http://www.google.co.jp/");
```

JavaScriptで何ができる？

- AndroidプログラムからJavaScriptコードを呼び出したり、逆にJavaScriptからAndroidのオブジェクトを操作したり関数を呼び出すことができる。
- 通常のWEBシステムのようにHTML側にJavaScriptを記述することにより、WEBページの動的書き換え等が可能。

例えば

JavaScript→Android

- ・変数を操作したり
- ・Android内のコンタクトリストをWEBページに表示したり
- ・GPSデータを取得したり
- ・取得したデータをSQLiteに格納してアプリに使ったり
- ・JavaScriptから電話をかけてみたり
- ・Androidアプリのタイトルを変更してみたり

etc...

JavaScriptと連携することにより、リモートサーバ側のロジックでAndroid内のデータを簡単に操作できる！

Android→JavaScript

- ・Android端末のメールや電話の着信を察知して、開いているWEBページに着信を表示するなど也能する！？
- ・HTMLの動的書き換えや入力フォームの自動補完等、リッチクライアントの提供(これはAndroidに限らず一般的なJavaScriptの用途)

Androidから利用した場合の差分

- Androidの標準ブラウザからでもPCブラウザからWEBページを見た場合と同等のことができるのかは現在調査中。

※予想としては、「Safari」、「Lunascript」と言ったWebkit使用ブラウザと同様の動きをすると思われる。

実際に動かしてみる1

◆ JavaScript側から電話をかけてみる

- ・WebViewを使うにはまずWebViewを“layout/main.xml”に定義。

```
<LinearLayout xmlns:android="..."  
    <WebView android:id="@+id/browser"  
        android:layout_width="wrap_content"  
        android:layout_height="120px" />  
</LinearLayout>
```

- ・onCreateでWebKitのViewを取得。

```
WebView browser = (WebView) findViewById(R.id.browser);
```

実際に動かしてみる2

- JavaScriptとAndroidをブリッジする。
(JavaオブジェクトとJavaScriptオブジェクトを繋ぐ)

```
JsObj jo = new JsObj(this);  
browser.addJavascriptInterface(jo, "Android")
```

→これでJsObjのcallPhone関数がJavaScript側ではAndroid.callPhone関数として呼び出せる。JavaScriptからAndroidのメソッドを呼び出すための作業であり、その逆は関係がない。

JSの使用OK

- JavaScriptの利用許可を与える

```
browser.getSettings().setJavaScriptEnabled(true);
```

- loadUrlでJavaScriptを記述したHTMLを読み込む

```
browser.loadUrl("http://xxxxxxxx.xx.xxxx/xxxx");
```

実際に動かしてみる3

- ・LoadUrlで読み込まれたHTML内に記述されたJavaScriptのイベントハンドラに従って関数が実行される。(呼び出せるのはpublic関数のみ、private関数を呼び出すとエラーになる。)

ブリッジしたAndroidオブジェクトのcallPhone関数を呼び出し電話をかける。

```
<script type="text/javascript">
```

```
...
```

```
document.frm1.btn1.onclick = function(){
```

```
    Android.callPhone(text);
```

```
}
```

```
...
```

```
</script>
```

btn1押下時に“text”の電話番号に電話をかける

JavaScriptの危険性

JavaScriptは本来Webページに動きを与えたり、ユーザビリティの向上に利用されるものであるが、悪意のある人間が利用することの危険性も指摘されている。

- HTML内に悪意あるJavaScriptを記述することで、キーロガー的なプログラムやクッキー情報を盗み出す、といったことが可能になり、個人情報や機密情報などが盗み取られる可能性がある。(→クロスサイトスクリプティング)
- 任意のコードが自由に実行できるということは、応用次第で様々な可能性があるが、反面こうした脆弱性に開発者は常に気を使う必要がある。

クロスサイトスクリプティング (Cross Site Scripting)

Webサイト利用者の入力した文字列がそのまま表示されるサービス(掲示板や確認メッセージ等)にJavaScriptなどのクライアントサイドスクリプトを仕込むことで成立する攻撃手段、またはその脆弱性。サーバサイドではなく利用者のブラウザ上で不正なプログラムが実行される点
が特徴。

Cascading Style Sheetsとの混同を避けるために通常XSSと略される。
攻撃者が対象となるサイトとは異なるサイトからスクリプトを送り込み訪問者に実行せしめることからクロスサイトと呼ばれる。

攻撃対象は脆弱サイトに訪問している利用者であり、cookie情報の盗難、Webページの改竄、マルウェアのダウンロード等が主な目的として行われる。

cookie情報内に重要な個人情報that格納されるサイトの場合(そうしたサイトにも問題があるが)、ログイン情報を乗っ取られたりクレジットカードの番号が抜かれたりといった危険性も...

それぞれが行うべき対策

利用者側	<p>信頼できるサイト以外はブラウザの設定でアクティブスクリプトをオフにする。Firefoxならば「NoScript」というアドオンがサイト毎にオンオフを指定できて便利。</p> <p>とはいえ、どこのサイトが信用できるかなど利用者側にはわからないので、結局できることは余りない。</p>
WEB管理者側	<p>掲示板などユーザーの入力がそのまま出力されるような画面では、出力時にメタ文字のエスケープを適切に行うことが最も重要となる。一度全てを抑制してから安全とわかったものだけ許可を出すホワイトリスト方式がセキュリティの基本。</p>
Androidアプリ開発者側	<p>端末内の任意のフォルダに自由にアクセスできるメソッドを作らないなど、例え悪意あるJavaScriptが実行されても重要な個人情報が流出しないような作りを心がける。</p>

まとめ

- AndroidでJavaScriptは利用可能。
- Java \leftrightarrow JavaScriptの相互の呼び出しができる。
- XSS等、JavaScriptを悪用した攻撃が可能なので、セキュリティには気を付ける必要がある。
- GoogleMAPS等のGoogleAPIもJavaScriptで利用できる。

参考URL

<http://chephes.cocolog-nifty.com/blog/2007/12/androidjavascriptdf21.html>

http://www.plants-web.jp/flashmind/blog/2007/12/android_webkit_javascript.html

http://www.adamrocker.com/blog/172/javascript_android_bridge.html

<http://www.adamrocker.com/blog/?s=addjavascriptinterface>

http://blog.livedoor.jp/de_colin/archives/1286650.html