自Android 4.4起，Android中的WebView开始基于Chromium（这大概是因为Android部门负责人从Andy Rubin变成了Chrome部门的主管Sundar Pichai了吧，^_^）。

这个改变使得WebView的性能大幅度提升，并且对HTML5, CSS3, and JavaScript有了更好的支持。

那么，作为一个客户端开发者，我们写代码的时候需要注意哪些呢？

一、多线程

如果你在子线程中调用WebView的相关方法，而不在UI线程，则可能会出现无法预料的错误。

所以，当你的程序中需要用到多线程时候，也请使用 runOnUiThread()方法来保证你关于WebView的操作是在UI线程中进行的：

```
runOnUiThread(newRunnable(){
    @Override
    publicvoid run(){
       // Code for WebView goes here
    }
});
```

二、线程阻塞

永远不要阻塞UI线程，这是开发Android程序的一个真理。虽然是真理，我们却往往不自觉的犯一些错误违背它，一个开发中常犯的错误就是：在UI线程中去等待JavaScript 的回调。

例如：

```
// This code is BAD and will block the UI thread
webView.loadUrl("javascript:fn()"); while(result ==null){
Thread.sleep(100); }
```

千万不要这样做，Android 4.4中，提供了新的Api来做这件事情。
evaluateJavascript() 就是专门来异步执行JavaScript代码的。

三、evaluateJavascript() 方法

专门用于异步调用JavaScript方法，并且能够得到一个回调结果。

示例：

```
mWebView.evaluateJavascript(script, new ValueCallback<String>() {
    @Override
    public void onReceiveValue(String value) {
         //TODO
    }
});
```

四、处理 WebView 中 url 跳转

新版WebView对于自定义scheme的url跳转，新增了更为严格的限制条件。
当你实现了 shouldOverrideUrlLoading() 或 shouldInterceptRequest() 回调，WebView 也只会在跳转url是合法Url时才会跳转。

例如，如果你使用这样一个url：

```
<ahref="showProfile"]]>Show Profile</a>
```

shouldOverrideUrlLoading() 将不会被调用。

正确的使用方式是：

```
<ahref="example-app:showProfile"]]>Show Profile</a>
```

对应的检测Url跳转的方式：

```
// The URL scheme should be non-hierarchical (no trailing slashes)
privatestaticfinalString APP_SCHEME ="example-app:";

@Override publicboolean shouldOverrideUrlLoading(WebView view,String
url){
    if(url.startsWith(APP_SCHEME)){
        urlData =URLDecoder.decode(url.substring(APP_SCHEME.length()),"UTF-8");
        respondToData(urlData);
        returntrue;
    }
    returnfalse; }
```

当然，也可以这样使用：

```
webView.loadDataWithBaseURL("example-app://example.co.uk/", HTML_DATA,
 null,"UTF-8",null);
```

五、UserAgent 变化

如果你的App对应的服务端程序，会根据客户端传来的UserAgent来做不同的事情，那么你需要注意的是，新版本的WebView中，UserAgent有了些微妙的改变：

```
Mozilla/5.0 (Linux; Android 4.4; Nexus 4 Build/KRT16H)
AppleWebKit/537.36(KHTML, like Gecko) Version/4.0 Chrome/30.0.0.0
Mobile Safari/537.36
```

使用 getDefaultUserAgent()方法可以获取默认的UserAgent，也可以通过：

```
mWebView.getSettings().setUserAgentString(ua);
mWebView.getSettings().getUserAgentString();
```

来设置和获取自定义的UserAgent。

六、使用addJavascriptInterface()的注意事项

从Android4.2开始。
只有添加 @JavascriptInterface 声明的Java方法才可以被JavaScript调用，例如：

```
class JsObject {
    @JavascriptInterface
    public String toString() { return "injectedObject"; }
 }
 webView.addJavascriptInterface(new JsObject(), "injectedObject");
 webView.loadData("", "text/html", null);
 webView.loadUrl("javascript:alert(injectedObject.toString())");
```

七、Remote Debugging

新版的WebView还提供了一个很厉害的功能：使用Chrome来调试你运行在WebView中的程序。