

文章分类

- 自定义View原理&应用系列 (10)
- Android基础 (17)
- Android常用汇总 (30)
- 最易懂的设计模式解析 (12)
- 源码分析 (7)
- 常用开源库 (6)
- 多线程汇总 (4)

阅读排行

- Android：你要的WebVi (25971)
- Android：最全面的 Web (25173)
- Android：你不知道的 W (21691)
- Android事件分发机制详 (17127)
- 这是一份很有诚意的 Pro (14826)
- 快来看看Google出品的P (13161)
- 手把手教你写一个完整的 (12961)
- Android开源库V - Layout (11618)
- Android消息推送：手把 (11038)
- Android开发：最全面、 (10565)



1. 交互方式总结

Android与js通过WebView互相调用方法，实际上是：

- Android去调用JS的代码
- JS去调用Android的代码

二者沟通的桥梁是WebView

对于android调用JS代码的方法有2种：

- 通过 WebView 的 loadUrl ()
- 通过 WebView 的 evaluateJavascript ()

对于JS调用Android代码的方法有3种：

- 通过 WebView 的 addJavascriptInterface () 进行对象映射
- 通过 WebViewClient 的 shouldOverrideUrlLoading () 方法回调拦截 url
- 通过 WebChromeClient 的 onJsAlert()、onJsConfirm()、onJsPrompt () 方法回调拦截JS对话框 alert()、confirm()、prompt () 消息

2. 具体分析

2.1 Android通过WebView调用 JS 代码

对于Android调用JS代码的方法有2种：

- 通过 WebView 的 loadUrl ()
- 通过 WebView 的 evaluateJavascript ()

方式1：通过 WebView 的 loadUrl ()

- 实例介绍：点击Android按钮，即调用WebView JS (文本名为 javascript)中callJS ()
- 具体使用：

步骤1：将需要调用的JS代码以 .html 格式放到src/main/assets文件夹里

- 为了方便展示，本文是采用Andorid调用本地JS代码说明；
- 实际情况时，Android更多的是调用远程JS代码，即将加载的JS代码路径改成url即可

需要加载JS代码：JavaScript.html

```
1 // 文本名: javascript
2 <!DOCTYPE html>
3 <html>
4
5   <head>
6     <meta charset="utf-8">
```

```

7         <title>Carson_Ho</title>
8
9         // JS代码
10        <script>
11        // Android需要调用的方法
12        function callJS() {
13            alert("Android调用了JS的callJS方法");
14        }
15    </script>
16
17    </head>
18
19    </html>

```

步骤2：在Android里通过WebView设置调用JS代码

Android代码：MainActivity.java

注释已经非常清楚

```

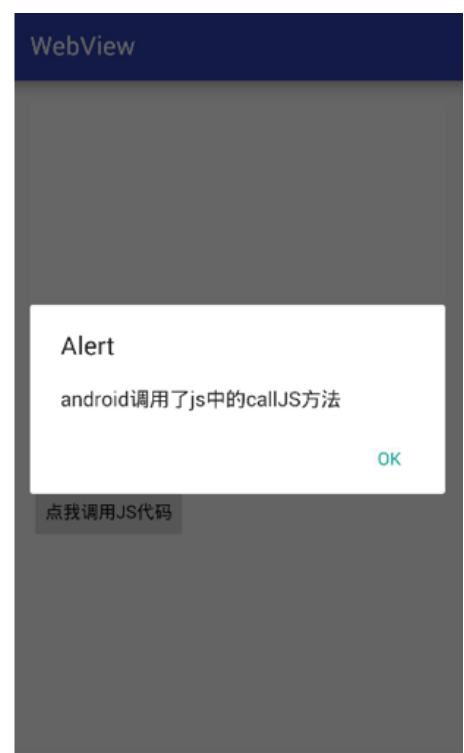
1    public class MainActivity extends AppCompatActivity {
2
3        WebView mWebView;
4        Button button;
5
6        @Override
7        protected void onCreate(Bundle savedInstanceState) {
8            super.onCreate(savedInstanceState);
9            setContentView(R.layout.activity_main);
10
11            mWebView = (WebView) findViewById(R.id.webview);
12
13            WebSettings webSettings = mWebView.getSettings();
14
15            // 设置与Js交互的权限
16            webSettings.setJavaScriptEnabled(true);
17            // 设置允许JS弹窗
18            webSettings.setJavaScriptCanOpenWindowsAutomatically(true);
19
20            // 先载入JS代码
21            // 格式规定为:file:///android_asset/文件名.html
22            mWebView.loadUrl("file:///android_asset/javascript.html");
23
24            button = (Button) findViewById(R.id.button);
25
26
27            button.setOnClickListener(new View.OnClickListener() {
28                @Override
29                public void onClick(View v) {
30                    // 必须另开线程进行JS方法调用(否则无法调用)
31                    mWebView.post(new Runnable() {
32                        @Override
33                        public void run() {
34
35                            // 注意调用的JS方法名要对应上
36                            // 调用javascript的callJS()方法
37                            mWebView.loadUrl("javascript:callJS()");
38                        }
39                    });
40
41                }
42            });
43
44            // 由于设置了弹窗检验调用结果,所以需要支持js对话框
45            // webview只是载体, 内容的渲染需要使用webviewChromClient类去实现
46            // 通过设置WebChromeClient对象处理JavaScript的对话框
47            // 设置响应js 的Alert() 函数
48            mWebView.setWebChromeClient(new WebChromeClient() {
49                @Override

```

```

50         public boolean onJsAlert(WebView view, String url, String message, final JsResult result) {
51             AlertDialog.Builder b = new AlertDialog.Builder(MainActivity.this);
52             b.setTitle("Alert");
53             b.setMessage(message);
54             b.setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener() {
55                 @Override
56                 public void onClick(DialogInterface dialog, int which) {
57                     result.confirm();
58                 }
59             });
60             b.setCancelable(false);
61             b.create().show();
62             return true;
63         }
64     }
65 });
66
67
68 }
69 }

```



特别注意：JS代码调用一定要在 `onPageFinished()` 回调之后才能调用，否则不会调用。

`onPageFinished()` 属于 `WebViewClient` 类的方法，主要在页面加载结束时调用

方式2：通过WebView的 `evaluateJavascript()`

- 优点：该方法比第一种方法效率更高、使用更简洁。

1. 因为该方法的执行不会使页面刷新，而第一种方法（`loadUrl()`）的执行则会。
2. Android 4.4 后才可使用

- 具体使用

```

1 // 只需要将第一种方法的loadUrl()换成下面该方法即可
2 mWebView.evaluateJavascript("javascript:callJS()", new ValueCallback<String>() {
3     @Override
4     public void onReceiveValue(String value) {
5         //此处为 js 返回的结果

```

```
6         }
7     });
8 }
```

2.1.2 方法对比

调用方式	优点	缺点	使用场景
使用loadUrl ()	方便简洁	效率低; 获取返回值麻烦	不需要获取返回值, 对性能要求较低时
使用evaluateJavascript ()	效率高	向下兼容性差 (仅Android 4.4以上可用)	Android 4.4以上

2.1.3 使用建议

两种方法混合使用，即Android 4.4以下使用方法1，Android 4.4以上方法2

```
1 // Android版本变量
2 final int version = Build.VERSION.SDK_INT;
3 // 因为该方法在 Android 4.4 版本才可使用，所以使用时需进行版本判断
4 if (version < 18) {
5     mWebView.loadUrl("javascript:callJS()");
6 } else {
7     mWebView.evaluateJavascript ("javascript:callJS()", new ValueCallback<String>() {
8         @Override
9         public void onReceiveValue(String value) {
10             //此处为 js 返回的结果
11         }
12     });
13 }
```

2.2 JS通过WebView调用 Android 代码

对于JS调用Android代码的方法有3种：

- 1. 通过 WebView 的 addJavascriptInterface () 进行对象映射
- 2. 通过 WebViewClient 的 shouldOverrideUrlLoading () 方法回调拦截 url
- 3. 通过 WebChromeClient 的 onJsAlert ()、 onJsConfirm ()、 onJsPrompt () 方法回调拦截JS对话框 alert ()、 confirm ()、 prompt () 消息

2.2.1 方法分析

方式1：通过 WebView 的 addJavascriptInterface () 进行对象映射

步骤1：定义一个与JS对象映射关系的Android类：AndroidtoJs

AndroidtoJs.java (注释已经非常清楚)

```
1 // 继承自Object类
2 public class AndroidtoJs extends Object {
3
4     // 定义JS需要调用的方法
5     // 被JS调用的方法必须加入@JavascriptInterface注解
6     @JavascriptInterface
7     public void hello(String msg) {
8         System.out.println("JS调用了Android的hello方法");
9     }
10 }
```

步骤2：将需要调用的JS代码以 .html 格式放到src/main/assets文件夹里

需要加载JS代码： javascript.html

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <title>Carson</title>
6      <script>
7
8
9      function callAndroid() {
10         // 由于对象映射，所以调用test对象等于调用Android映射的对象
11         test.hello("js调用了android中的hello方法");
12     }
13   </script>
14 </head>
15 <body>
16   //点击按钮则调用callAndroid函数
17   <button type="button" id="button1" onclick="callAndroid()"></button>
18 </body>
19 </html>
```

步骤3：在Android里通过WebView设置Android类与JS代码的映射

详细请看注释

```
1  public class MainActivity extends AppCompatActivity {
2
3      WebView mWebView;
4
5      @Override
6      protected void onCreate(Bundle savedInstanceState) {
7          super.onCreate(savedInstanceState);
8          setContentView(R.layout.activity_main);
9
10         mWebView = (WebView) findViewById(R.id.webview);
11         WebSettings webSettings = mWebView.getSettings();
12
13         // 设置与Js交互的权限
14         webSettings.setJavaScriptEnabled(true);
15
16         // 通过addJavascriptInterface() 将Java对象映射到JS对象
17         //参数1: Javascript对象名
18         //参数2: Java对象名
19         mWebView.addJavascriptInterface(new AndroidtoJs(), "test");//AndroidtoJS类对象映射
20
21         // 加载JS代码
22         // 格式规定为:file:///android_asset/文件名.html
23         mWebView.loadUrl("file:///android_asset/javascript.html");
```

WebView

点击调用Android代码

```
03-04 02:37:42.172 6904-6960/scut.carson_ho.webview I/System.out: JS调用了Android的hello方法
```

特点

- 优点：使用简单

仅将Android对象和JS对象映射即可

- 缺点：存在严重的漏洞问题，具体请看文章：[你不知道的 Android WebView 使用漏洞](#)

方式2：通过 WebViewClient 的方法 shouldOverrideUrlLoading () 回调拦截 url

- 具体原理：

1. Android通过 WebViewClient 的回调方法 shouldOverrideUrlLoading () 拦截 url
2. 解析该 url 的协议
3. 如果检测到是预先约定好的协议，就调用相应方法

即JS需要调用Android的方法

- 具体使用：

步骤1：在JS约定所需要的Url协议

JS代码：javascript.html

以.html格式放到src/main/assets文件夹里

```
1 <!DOCTYPE html>
2 <html>
3
4 <head>
5 <meta charset="utf-8">
```

```

6         <title>Carson_Ho</title>
7
8         <script>
9             function callAndroid() {
10                 /*约定的url协议为: js://webview?arg1=111&arg2=222*/
11                 document.location = "js://webview?arg1=111&arg2=222";
12             }
13         </script>
14     </head>
15
16     <!-- 点击按钮则调用callAndroid () 方法 -->
17     <body>
18         <button type="button" id="button1" onclick="callAndroid()">点击调用Android代码</button>
19     </body>
20 </html>

```

当该JS通过Android的 `mWebView.loadUrl("file:///android_asset/javascript.html")` 加载后, 就会回调 `shouldOverrideUrlLoading ()`, 接下来继续看步骤2:

步骤2: 在Android通过WebViewClient复写 `shouldOverrideUrlLoading ()`

MainActivity.java

```

1 public class MainActivity extends AppCompatActivity {
2
3     WebView mWebView;
4     // Button button;
5
6     @Override
7     protected void onCreate(Bundle savedInstanceState) {
8         super.onCreate(savedInstanceState);
9         setContentView(R.layout.activity_main);
10
11         mWebView = (WebView) findViewById(R.id.webview);
12
13         WebSettings webSettings = mWebView.getSettings();
14
15         // 设置与Js交互的权限
16         webSettings.setJavaScriptEnabled(true);
17         // 设置允许JS弹窗
18         webSettings.setJavaScriptCanOpenWindowsAutomatically(true);
19
20         // 步骤1: 加载JS代码
21         // 格式规定为:file:///android_asset/文件名.html
22         mWebView.loadUrl("file:///android_asset/javascript.html");
23
24
25         // 复写WebViewClient类的shouldOverrideUrlLoading方法
26         mWebView.setWebViewClient(new WebViewClient() {
27             @Override
28             public boolean shouldOverrideUrlLoading(WebView view
29
30                 // 步骤2: 根据协议的参数, 判断是否是所需要的url
31                 // 一般根据scheme (协议格式) & authority (协议:
32                 // 假定传入进来的 url = "js://webview?arg1=111&a
33
34                 Uri uri = Uri.parse(url);
35                 // 如果url的协议 = 预先约定的 js 协议
36                 // 就解析往下解析参数
37                 if (uri.getScheme().equals("js")) {
38
39                     // 如果 authority = 预先约定协议里的 webvie
40                     // 所以拦截url, 下面JS开始调用Android需要的方
41                     if (uri.getAuthority().equals("webview")) {
42
43                         // 步骤3:
44                         // 执行JS所需要调用的逻辑
45                         System.out.println("js调用了Android的方
46                         // 可以在协议上带有参数并传递到Android上

```

关闭


```
47                                     HashMap<String, String> params = new Has
48                                     Set<String> collection = uri.getQueryPar
49
50                                     }
51
52                                     return true;
53                                     }
54                                     return super.shouldOverrideUrlLoading(view, url)
55                                     }
56                                     }
57                                     );
58                                     }
59                                     }
```



特点

- 优点：不存在方式1的漏洞；
- 缺点：JS获取Android方法的返回值复杂。

如果JS想要得到Android方法的返回值，只能通过 WebView 的 `loadUrl()` 去执行 JS 方法把返回值传递回去，相关的代码如下：

```
1 // Android: MainActivity.java
2 mWebView.loadUrl("javascript:returnResult(" + result + ")");
3
4 // JS: javascript.html
5 function returnResult(result){
6     alert("result is" + result);
7 }
```

方式3：通过 WebChromeClient 的 `onJsAlert()`、`onJsConfirm()`、`onJsPrompt()` 方法回调拦截JS对话框 `alert()`、`confirm()`、`prompt()` 消息

在JS中，有三个常用的对话框方法：

方法	作用	返回值	备注
alert ()	弹出警告框	没有	在文本加入\n可换行
confirm ()	弹出确认框	两个返回值	<ul style="list-style-type: none">• 返回布尔值；• 通过该值可判断点击时确认还是取消：true表示点击了确认，false表示点击了取消。
prompt ()	弹出输入框	任意设置返回值	<ul style="list-style-type: none">• 点击“确认”：返回输入框中的值；• 点击“取消”：返回null。

方式3的原理：Android通过 WebChromeClient 的 onJsAlert() 、 onJsConfirm() 、 onJsPrompt () 方法回调分别拦截JS对话框（即上述三个方法），得到他们的消息内容，然后解析即可。

下面的例子将用拦截 JS的输入框（即 prompt () 方法）说明：

- 1. 常用的拦截是：拦截 JS的输入框（即 prompt () 方法）
- 2. 因为只有 prompt () 可以返回任意类型的值，操作最全面方便、更加灵活；而alert () 对话框没有返回值；confirm () 对话框只能返回两种状态（确定 / 取消）两个值

步骤1：加载JS代码，如下：

javascript.html

以.html格式放到src/main/assets文件夹里

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Carson_Ho</title>
6
7     <script>
8
9     function clickprompt() {
10      // 调用prompt ()
11      var result=prompt("js://demo?arg1=111&arg2=222");
12      alert("demo " + result);
13    }
14
15    </script>
16  </head>
17
18  <!-- 点击按钮则调用clickprompt() -->
19  <body>
20    <button type="button" id="button1" onclick="clickprompt()">点击调用Android代码</button>
21  </body>
22 </html>
```

当使用 mWebView.loadUrl("file:///android_asset/javascript.html") 加载了上述JS代码后，就会触发回调 onJsPrompt () ，具体如下：

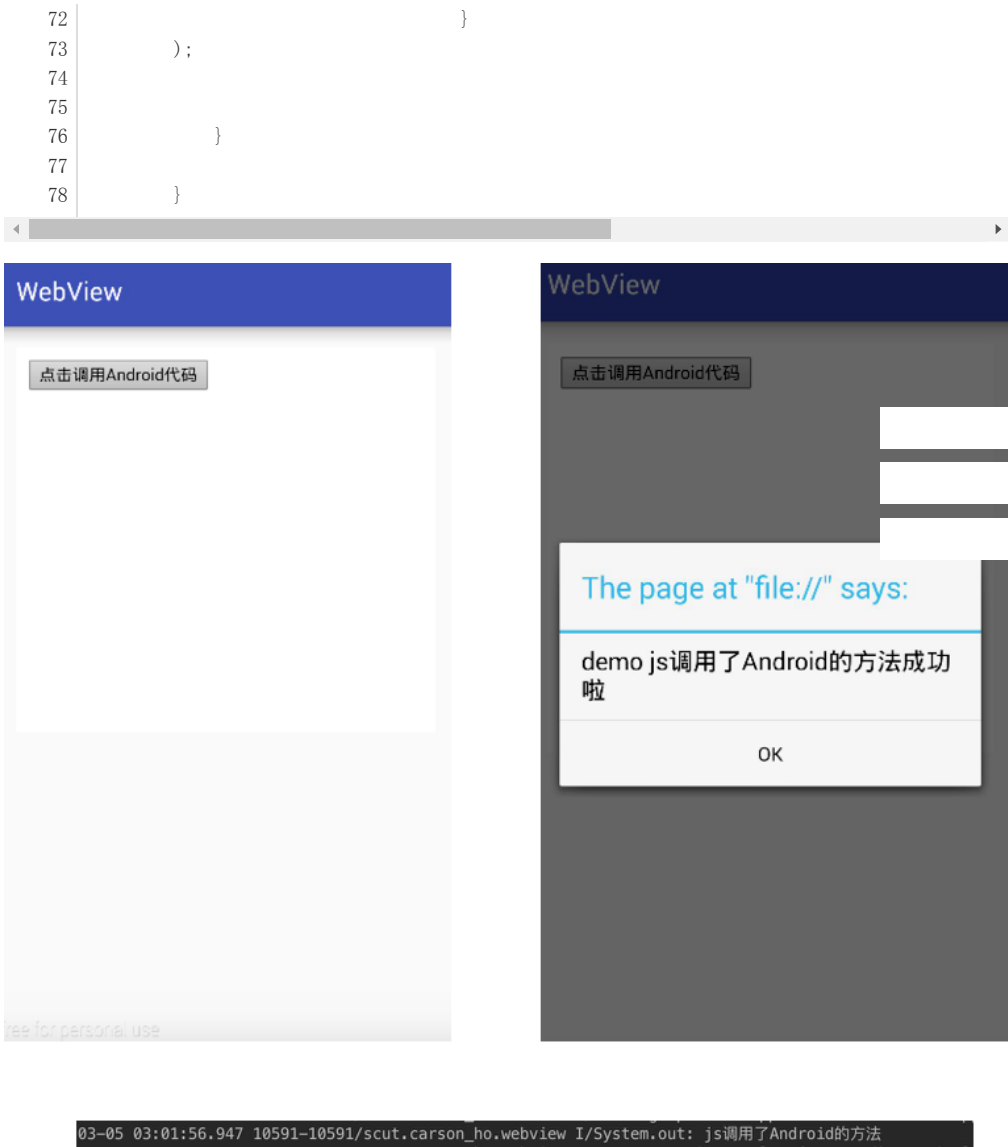
- 1. 如果是拦截警告框（即 alert() ），则触发回调 onJsAlert () ；
- 2. 如果是拦截确认框（即 confirm() ），则触发回调 onJsConfirm () ；

步骤2：在Android通过 WebChromeClient 复写 onJsPrompt ()

```

1 public class MainActivity extends AppCompatActivity {
2
3     WebView mWebView;
4     // Button button;
5
6     @Override
7     protected void onCreate(Bundle savedInstanceState) {
8         super.onCreate(savedInstanceState);
9         setContentView(R.layout.activity_main);
10
11         mWebView = (WebView) findViewById(R.id.webview);
12
13         WebSettings webSettings = mWebView.getSettings();
14
15         // 设置与Js交互的权限
16         webSettings.setJavaScriptEnabled(true);
17         // 设置允许JS弹窗
18         webSettings.setJavaScriptCanOpenWindowsAutomatically(true);
19
20         // 先加载JS代码
21         // 格式规定为:file:///android_asset/文件名.html
22         mWebView.loadUrl("file:///android_asset/javascript.html");
23
24
25         mWebView.setWebChromeClient(new WebChromeClient() {
26             // 拦截输入框(原理同方式2)
27             // 参数message:代表prompt () 的内容 (不是url)
28             // 参数result:代表输入框的返回值
29             @Override
30             public boolean onJsPrompt(WebView view, String url
31                 // 根据协议的参数,判断是否是所需要的url(原理
32                 // 一般根据scheme (协议格式) & authority (协
33                 //假定传入进来的 url = "js://webview?arg1=111&
34
35                 Uri uri = Uri.parse(message);
36                 // 如果url的协议 = 预先约定的 js 协议
37                 // 就解析往下解析参数
38                 if (uri.getScheme().equals("js")) {
39
40                     // 如果 authority = 预先约定协议里的 webv
41                     // 所以拦截url,下面JS开始调用Android需要的
42                     if (uri.getAuthority().equals("webview"))
43
44                         //
45                         // 执行JS所需要调用的逻辑
46                         System.out.println("js调用了Android的");
47                         // 可以在协议上带有参数并传递到Android
48                         HashMap<String, String> params = new HashMap<>();
49                         Set<String> collection = uri.getQueryParameters();
50
51                         //参数result:代表消息框的返回值(输入值)
52                         result.confirm("js调用了Android的方法");
53                     }
54                     return true;
55                 }
56                 return super.onJsPrompt(view, url, message, result);
57             }
58
59             // 通过alert()和confirm()拦截的原理相同,此处不作过多讲述
60
61             // 拦截JS的警告框
62             @Override
63             public boolean onJsAlert(WebView view, String url,
64                 return super.onJsAlert(view, url, message, result);
65             }
66
67             // 拦截JS的确认框
68             @Override
69             public boolean onJsConfirm(WebView view, String url,
70                 return super.onJsConfirm(view, url, message, result);
71             }

```



2.2.2 三种方式的对比 & 使用场景

调用方式	优点	缺点	使用场景
通过addJavascriptInterface () 进行添加对象映射	方便简洁	Android 4.2以下存在漏洞问题	Android 4.2以上相对简单互调场景
通过 WebViewClient 的方法shouldOverrideUrlLoading () 回调拦截 url	不存在漏洞问题	使用复杂：需要进行协议的约束；从 Native 层往 Web 层传值比较繁琐	不需要返回值情况下的互调场景 (iOS主要使用该方式)
通过 WebChromeClient 的onJsAlert(), onJsConfirm(), onJsPrompt () 方法回调拦截JS对话框消息	不存在漏洞问题	使用复杂：需要进行协议的约束；	能满足大多数情况下的互调场景

3. 总结

- 本文主要对**Android通过WebView与JS的交互方式进行了全面介绍**
- 关于WebView的系列文章对你有所帮助
 - Android开发：最全面、最易懂的Webview详解
 - Android：你不知道的 WebView 使用漏洞
 - 手把手教你构建 Android WebView 的缓存机制 & 资源预加载方案
- 接下来我会继续讲解其他安卓开发的知识，有兴趣可以继续关注Carson_Ho的**安卓开发笔记**！！！！

请帮顶和评论点赞！因为你们的赞同/鼓励是我写作的最大动力！

顶

45

踩

0

上一篇

Android：你不知道的 WebView 使用漏洞

下一篇

快来看看Google出品的Protocol Buffer，别只会用Json和XML了

相关文章推荐

参考知识库



Android知识库
34825 关注 | 3135 收录



JavaScript知识库
15343 关注 | 1516 收录



Java 知识库
27951 关注 | 3746 收录



Java EE知识库
19025 关注 | 1408 收录



Java SE知识库
26740 关注 | 578 收录



jQuery知识库
9397 关注 | 948 收录



AngularJS知识库
5022 关注 | 570 收录

猜你在找

- JavaScript大神之路第一季
- Struts2架构设计之路 自主编写Web开发框架
- Python大型网络爬虫项目开发实战（全套）
- 微信小程序开发实战第三季
- 2017软考软件设计师下午案例分析视频培训课程
- 2017软考系统集成项目管理工程师视频教程精讲 基础知识（上）
- 开源摄像机EasyCamera开发教程
- 2017软考系统集成项目管理工程师-上午历年真题解析培训视频课程
- 深度学习框架-Tensorflow案例实战视频课程
- 深度学习Caffe框架入门视频课程

查看评论

12楼

colorizedpanda


2017-05-26 18:23发表




为什么我用的前两种android调用js的方法不能显示提示框呢？
webview显示的是网页无法打开
位于file:///android_asset/javascript.html的网页无法加载,因为:
net:Err_FILE_NOT_FOUND
错误提示是:

05-26 18:12:12.740 31415-31460/com.lxy.webviewjsdemo E/ActivityThread: Failed to find provider info for com.google.settings
05-26 18:12:14.347 31415-31552/com.lxy.webviewjsdemo E/libEGL: validate_display:255 error 3008 (EGL_BAD_DISPLAY)
05-26 18:18:41.690 31415-31459/com.lxy.webviewjsdemo E/AndroidProtocolHandler: Unable to open asset URL: file:///android_asset/javascript.html

11楼 小玄鸡233 2017-05-11 00:41发表

 最近我遇到了js 在某个html 无法调用Android函数的问题，支用添加对象映射的方式，同一个接口 其他页面都能正常调用 楼主有遇到过么


10楼 qq_35248272 2017-05-08 14:19发表

 引用"m0_38013766"的评论：
0


9楼 qq_35248272 2017-05-08 14:10发表

 引用"qq_35248272"的评论：
引用"m0_38013766"的评论：
0
1
2


8楼 qq_35248272 2017-05-08 11:33发表

 引用"m0_38013766"的评论：
0
1

7楼 爱先森 2017-04-26 10:08发表

 "必须另开线程进行JS方法调用(否则无法调用)"？
为什么，我没有开新线程也调用成功了？

6楼 夜雨晨曦 2017-03-24 13:59发表

 ddddddddddddddddddd


5楼 林凡,,,, 2017-03-24 13:27发表

达瓦大阿瓦打我的娃

4楼 小得 2017-03-23 14:40发表

 mark
<https://www.exuetechnology.com>

3楼 naruto___ 2017-03-23 08:34发表

 楼主你好，你说的"那个要另开线程才能调用webview.loadUrl()"描述的可能有问题吧，应该是确保要在主线程中进行调用，用runOnUiThread()也可以，而且你用的post也不会另起线程，而是把消息添加到当前主线程中，让looper去执行，拙见，勿怪


2楼 zhadan_1987 2017-03-22 14:37发表

 js学习资料：
<http://www.codeyyy.com/javascript/index.html>


1楼 m0_38013766 2017-03-22 11:41发表

 0


Re: qq_35248272 2017-05-08 14:12发表

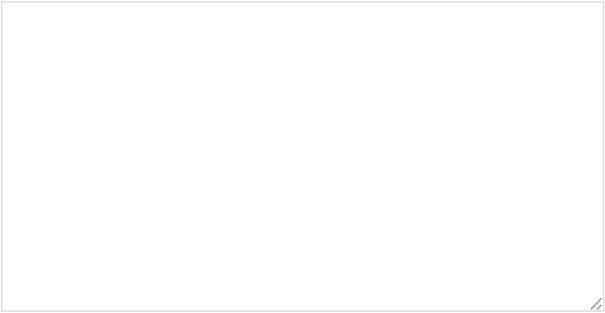
 回复m0_38013766 : 6666666

Re: qq_35248272 2017-05-08 14:12发表

 回复qq_35248272 : 555

发表评论

用户 名： xfhy_
评论 内容： 



提交

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker
OpenStack VPN Spark ERP IE10 Eclipse CRM JavaScript 数据库 Ubuntu ...
WAP jQuery BI HTML5 Spring Apache .NET API HTML SDK IIS Fedora
LBS Unity Splashtop UML components Windows Mobile Rails QEMU KDE Cassandra
CloudStack FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App
SpringSide Maemo Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP
HBase Pure Solr Angular Cloud Foundry Redis Scala Django Bootstrap

公司简介 | 招贤纳士 | 广告服务 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-660-0108 | 北京创新乐知信息技术有限公司 版权所有 | 江苏知之为计算机有限公司 |

江苏乐知网络技术有限公司

京 ICP 证 09002463 号 | Copyright © 1999-2017, CSDN.NET, All Rights Reserved

