WebView面试题

版本：2018/9/1-1

题目来源:

1. WebView和JS交互

目录

基本使用

1、WebView的基本使用?

> 1. webview.setWebViewClient(new WebViewClient());
> 2. webview.setWebChromeClient(new WebChromeClient());
> 3. webview.loadUrl("https://www.baidu.com/");
> 4. webview.requestFocus();
> 5. AndroidManifest.xml配置网络权限 `<uses-permission android:name="android.permission.INTERNET"/>`

交互

2、Android和JS的交互有哪些方式?

> 1. Android中调用JS中的函数
> 2. JS通过WebView调用Android代码

Android调用JS

3、Android中如何直接调用JS中的函数?

> 1. html中有js的方法callJs()里面去调用alert()方法
> 2. webview.setWebChromeClient()中的onJsAlert()去处理JS对话框
> 3. webview.loadUrl("javascript:callJS()")去直接调用

4、如何通过evaluateJavascript直接调用JS中的函数?

> 1. webview.evaluateJavascript("javascript:callJS()")去调用JS的方法，并且在ValueCallback中能获取JS的返回值。

5、建议混合使用上面的两种交互方法

> 1. API < 18, Android 4.4以下使用loadUrl
> 2. API >=18, 使用evaluateJavascript

JS调用Android

6、JS调用Android有哪些方法?

> 三种
>
> 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

7、addJavascriptInterface()对象映射是如何使用的?

> 1. html中调用某对象的某个方法：如 `jsObject.hello("js调用了android中的hello方法");`
> 2. Android中实现一个自定义类，具有方法hello()，该方法使用注解@JavascriptInterface
> 3. 进行映射 `webview.addJavascriptInterface(new 自定义类(), "jsObject")`
> 4. 加载js代码： `webview.loadUrl("file:///android_asset/jsCallAndroid.html");`

8、addJavascriptInterface()存在严重漏洞

> 1. Android 4.2(17)版本开始通过 `@JavascriptInterface` 规避该漏洞
> 2. 该漏洞会导致攻击者能执行任意Java对象的方法。

9、通过WebViewClient的 `shouldOverrideUrlLoading` 进行url拦截

> 1. js中约定url协议
> 2. Android中在 `shouldOverrideUrlLoading` 进行解析拦截

10、JS中如何获得Android方法执行的返回值?

> 1. 只能采用特殊方法去处理
> 2. js定义一个方法，如命名为 `returnResult(result)`
> 3. android中通过 `mWebView.loadUrl("javascript:returnResult(" + result + ")");` 将方法返回值作为参数传入进去。

11、如何通过WebChromeClient的 `onJsAlert()\onJsConfirm()\onJsPrompt()` 方法回调拦截JS对话框 `alert()\confirm\prompt()` 消息

> 1. Android中通过 `WebChromeClient` 的 `onJsAlert()\onJsConfirm()\onJsPrompt()` 方法回调分别拦截JS对话框。
> 2. 本质是JS弹出对画框，Android拦截了对话框，解析参数中的url等信息，去做事先约定好的任务。

12、三种JS调用Android代码的方法的总结

调用方法	优点	缺点	使用场景
addJavascriptInterface():对象映射	方便简洁	Android4.2以下存在漏洞	Android4.2以上
WebViewClient的shouldOverrideUrlLoading:拦截Url	不存在漏洞	使用复杂; 需要进行协议约束;	不需要返回值情况下的互动场景 （IOS中主要使用该方法）
WebChromeClient的onJsAlert等方法回调拦截Js的对话框消息	不存在漏洞	使用复杂; 需要进行协议约束;	能满足大多数情况下的互调场景

WebView的坑

13、addJavascriptInterface()的坑

> 1. API <= 16时，WebView.addJavascriptInterface()有安全漏洞

14、webview在布局文件中的使用

> 1. webview在布局文件中的使用：webview写在其他容器中时。终止时要先将WebView从容器中remove掉，再去执行WebView.removeAllViews()和WebView.destory()方法，能防止内存泄漏。

15、jsbridge的作用?

> 让js和android相互调用。
>
> 1. loadUrl()
> 2. evaluateJavascript()

3. addJavascriptInterface()
4. WebViewClientshouldOverrideUrlLoading()
5. WebChormeClient拦截JS对话框

16、webViewClient.onPageFinished()的坑

1. webview跳转各种url时，不断回调这个方法。
2. 建议使用 `WebChromeClient.onProgressChanged()` 方法比较好

17、WebView硬件加速导致页面渲染问题

1. 开启硬件加速会让加载更顺滑
2. 容易出现页面加载闪烁的问题。
3. 解决办法：暂时关闭WebView的硬件加速。

内存泄漏

18、WebView的内存泄露

1. WebView持有外部如Activity的引用，导致内存泄漏

19、内存泄漏解决办法

1. WebView用于独立线程：会涉及到IPC。但是简单粗暴，直接kill进程，能有效避免泄露。
2. 对传入WebView的Context使用弱引用，使用结束时，将WebView从父容器中remove后在进行清理工作。