基础环境准备：

1. 环境准备：安装 JDK并配置环境变量，安装 Android Studio (也有自带的 java 环境)
2. 新建项目或者通过 Git 导入原项目 (Project) ，打开 app 模块(module)
3. 自定义继承自系统 WebView 的类，设置基础属性
4. 新建 Activity，在布局中放入 WebView
5. 使用 WebView 的 loadUrl(url) 方法，加载目标链接

依赖管理：

1. Jar 包(aar 包)依赖：在 module gradle 文件 dependencies 中，添加 compile fileTree(include: ['*.jar'], dir: 'libs')，将所需 Jar 包放入 module 下 lib 文件夹内，然后 sync project with gradle files 让 jar 包可用；如果是 aar 包，需要额外在 build.gradle 添加 repositories{flatDir{dirs 'libs'} 和 dependencies{compile '包名:类库名:版本号@aar'}
2. gradle 依赖：类似 maven 依赖管理，compile 'groupId:artifactId:version'，例如 compile 'com.google.code.gson:gson:2.8.0'
3. module 依赖: 在项目目录下 import module，并在主 module gradle 文件中添加 compile project(':moduleName')，也可以手动将项目考到 project 根目录下，并手动在 setting.gradle 中 include module

Js 调用 Android 原生方法：

1. 通过 addJavascriptInterface 方式交互 - **推荐**
   首先，新建任意名称 Java 类，例如 JsNotify 类，内部任意定义 public 类型方法，并标注注解 @JavascriptInterface
   其次，给 WebView addJavascriptInterface(new JsNotify(), "javaObj");
   最后，Js 即可通过 javaObj 对象调用 JsNotify 类内带有 @JavascriptInterface 注解的任意 public 方法
   注意：Android 6.0 之后，需要动态申请权限，所以当调用的功能使用到敏感权限时，需要在原生方法里处理好申请权限逻辑，例如需要打开相机时，要申请相机权限，涉及到文件读写时要申请文件读写权限；
2. 通过 WebViewClient 的 shouldOverrideUrlLoading (webview, url) 回调拦截特征 url
   通过解析 Uri，拦截符合特征的请求，处理完之后，如果 retrun true，即可终止原流程，return false 仍会走原加载流程
3. 通过 WebChromeClient onJsAlert()、onJsConfirm()、onJsPrompt()方法回调拦截 JS 对话框消息
   通过解析 Uri，拦截符合特征的请求，处理完之后，如果 retrun true，即可终止原流程，return false 仍会走原加载流程

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 层传递值比较繁琐	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

Android 调用 Js 方法：

1. 通过 WebView 的 loadUrl(url) 方法
   加载目标页面，如果目标页面包含 callJS 的 js 方法，则可通过 WebView.loadUrl("javascript:callJS()") 方法调用；
   注意： JS代码调用一定要在 onPageFinished （） 回调之后才能调用，否则不会生效。
2. 通过 WebView.evaluateJavascript（"javascript:callJS()", valueCallBack）方法
   使用方法及注意事项和第一种方式相同，可通过 valueCallBack 获取返回值

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

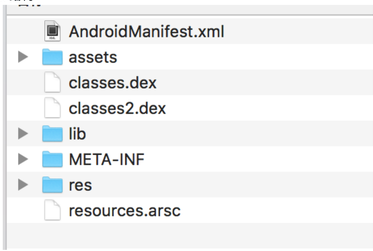
其他：

1. 网络权限等基础权限处理
2. 目前雁阵审批项目使用的 WebView 内核是腾讯 X5 内核，使用方式和系统内核基本一致，如果使用过程中遇到问题，可以从 X5 内核的方向找些资料参考下
3. 关于通过 WebView 实现 Js 和 Android 原生方法交互的详细说明及实例，可以参考这篇优秀博文

问题：

1.安卓的开发环境搭建。安卓系统5.X时对应的打apk的包怎么打？
build->Generate signed apk，选择签名，最后一步 signature version 需同时勾选 v1 和 v2，v2 在 7.0 以下版本无法使用；打完包之后即可发布，也可加固后发布。

2.打包代码的结构及其对应的含义？
项目结构，参考项目
apk 结构



3.打包时怎么更换url地址、图标？
url 在常量类中，图标在 drawable 或者 mipmap 中，图标通过 manifest 文件 application 节点 icon 属性配置

4.js调安卓原生代码的逻辑。比如、下拉刷新、导航条、调用相机、蓝牙、返回上页。退出保持登录状态。
Android 系统没有提供默认下拉刷新、导航条等功能，需要自己实现
打开蓝牙、相机等系统功能在回调里实现即可
保持登录状态需要在 H5 登陆之后，将保持登录状态的 cookie 存储到本地，再次进入时由 WebView 管理

5.安卓集成第三方的插件，比如、支持文档阅读。
参考依赖管理，依赖相关组件即可；目前雁阵审批里，支持的PDF、word 等文档阅读使用的是 X5 的插件，由 WebView 内核自动管理

6.开机即时启动应用的功能。--待实现 app设置为系统的launcher
可以监听开机事件广播，但是目前主流机型已不支持该广播，很难实现开机自启

7.自动获取拍照权限、拍照功能，压缩上传的功能。—待实现
Android 6.0 以上需要动态申请权限，需要用户明确点击授权，6.0 之前只需在 manifest 清单文件中声明需要拍照权限即可；
拍照、裁剪图片可以使用系统应用，可以参考这里；
压缩、上传需要自己实现

8.集成第三方插件的步骤及方式
参考依赖管理