混合开发Cordova介绍

- Cordova是一个混合开发打包工具，许多混合开发UI库都基于Cordova。将网页文件放到指定目录，运行一条命令，便可打包成Apk或webview。

安装环境

> 需要安装打包工具cordova, andriod环境，java

- 安装cordova: npm install -g cordova
- 安装andriod studio下载地址
  - 设置ANDROID_HOME为环境变量
  - 把Android SDK's的tools, tools/bin, platform-tools添加到PATH中
  - 打开Android studio，Tools->AVD Manager，新建一个虚拟安卓手机，以供调试
- 安装jdk下载地址
  - 设置JAVA_HOME为环境变量
- 设置环境变量和PATH后需要重启电脑。

Cordova 文档

```
> Cordova是一个打包工具。能把前端的HTML, CSS, JS打包成APK或webview.打包成APK的话可直接安装使用。
```

1. 先开发前端，在index.html中需添加

```
<script type="text/javascript" src="cordova.js">,
```

2. 新建项目：cordova create hello com.example.hello HelloWorld
3. 把网页的HTML, CSS, JS放到www文件夹中。
4. cordova run android把文件打包成apk，如果打开了安卓虚拟机，会自动安装。cordova run android --device，打包并自动部署到手机上，手机需通过数据线连接电脑。

部署成webView

需要安卓同事看看此页文档教程。
文中称在/framework目录下生成jar，指nodes_modules下cordova-android的framework目录详见

调试

- 安卓手机需在开发者选项中打开**允许调试**，并用数据线连接电脑，
- 在chrome://inspect中，可以看到手机和安卓虚拟机中打开的网页以及webview，点击inspect后调试

通过Cordova调用原生能力 文档

- 以查看电池为例

1. 先安装插件 cordova plugin add cordova-plugin-battery-status
2. 然后

```
window.addEventListener("batterystatus", onBatteryStatus, false);
function onBatteryStatus(status) {
     console.log("Level: " + status.level + " isPlugged: " + status.isPlugged);
}
```

3. 更多功能详见文档

Cordova生命周期

- devicereadys事件。Cordova加载完成时触发，所以前端代码需要监听到此事件后调用

```
document.addEventListener("deviceready", onDeviceReady, false);
function onDeviceReady() {
// Now safe to use device APIs
}
```

- pause事件。App进入后台时触发，通常表示用户切换到了其他程序。
- resume事件。表示进入后台的App又被打开了。
- 还有一些监听手机按钮的事件，点击阅读

最佳实践

- 采用SPA。

> Cordova app需等待deviceReady事件。若不采用SPA,不仅需下载新文件，还得再次等待这个事件

- 用touch，不用click

> 为了与touch和touch hould事件区分开来，click会有300ms延迟

- 制作动画用transition，不要操作dom
- 移动端网络情况较差，多用本地存储，谨慎传递数据

> 深入阅读
> 1. "You half assed it"
> 2. "Top Ten Performance Tips for PhoneGap and Hybrid Apps"
> 3. "Fast Apps and Sites with JavaScript"

- 妥善处理网络联通和断开的情况。而且NetWork connection API不靠谱。需发请求来判断网络是否畅通。

> 深入阅读："Is This Thing On?"

开发中会碰到需要解决的细节

> 开发前需要后台、IOS、安卓、前端要一起商量规划好的细节问题

js怎么调用Native里的方法，Native怎么调用js里的方法

- webview里通过Url Schema通知Native调用方法
  - Native监听Url Schema的变化，兼容性好，不存在等待webview加载的问题
- webview里直接调用Native注入的方法
  - 因为是注入的方法，有一个生命周期，可能会出现js代码已经都实例化完毕，而方法还没被注入的问题。

跳转

- Native跳H5
- H5跳Native
- H5跳H5（这里还要分内嵌的场景）

- H5新开Webview打开H5

跳转时的动画设计

要仔细管理history，以免出现死循环

Header组件的设计

> header组件需用Native组件，因为：
> - 其它主流容器都是这么做的，比如微信、手机百度、携程
> - 没有header一旦网络出错出现白屏，APP将陷入假死状态，这是不可接受的，而一般的解决方案都太业务了

防止假死

- 使用Native的Header组件，就是为了防止假死
- 通过检测网络情况，webview里js代码执行情况，给Header的后退按钮注册不同的方法，来避免假死。

webview里如何发请求

- webview打开在线站点时，用ajax发请求
- webview打开本地网站时
  - 调用native注入的方法发请求
  - 使用url schema，native监测到url schema，然后发请求，并执行js的回调

账号系统的设计

webview和native如何同步登陆状态。

- 为简化逻辑，请求需要都通过native发出
- 登陆和登出，也需要通过调用统一的组件。

账号信息获取

增量机制：即在线更新混H5写的模块

1. 给每个模块一个版本号
2. app每次启动时军询问后台是否有更新
3. 有更新则下载新的资源完成更新

NativeUI

- 可以封装一些非常通用的NativeUI以供调用

静态资源管理

> 因为开发时宿主环境是浏览器，上线后是webview，静态资源的位置可能会变化

- 静态资源都读取线上资源，Native可拦截webview请求，实现一些缓存的逻辑。

好文章

1. 浅谈Hybrid技术的设计与实现