目　　录

目前主流APP的种类

谁能说一下我们公司的APP用了哪些模式

目前主流的app大致分为三种：

- **Native APP**

Native App是一种基于智能手机本地操作系统如IOS、Android、WP并使用原生程式编写运行的第三方应用程序,也叫本地app。

- **Web App(HTML5)**

WebApp是指基于Web的系统和应用，其作用是向广大的最终用户发布一组复杂的内容和功能。

- **Hybrid App**

Hybrid App同时使用网页语言与程序语言开发，通过应用商店区分移动操作系统分发，用户需要安装使用的移动应用

一张图看懂app

Native App

优点：

- 能够访问手机的所有功能（GPS，相机等）
- 更好的运行速度、性能和总体的用户体验
- 支持离线工作（因为是在设备上运行而非Web）
- 支持丰富的图形和动画
- 在应用商店轻易地找到应用并且在主屏幕上能轻易找到应用图标

缺点：

- 开发成本较高
- 范围限制较多（只能访问在特定操作系统上运行的设备）
- 用户必须手动下载更新最新版本
- 内容限制（应用商店限制）

Yellow Pages Group

Web App

优点：

- 适用范围广（覆盖所有智能手机）
- 开发成本较低
- 方便、快捷地部署（无需提交到应用商店）
- 无内容限制
- 用户总能访问到最新版本（没有手动更新需求）

缺点：

- 较差的和较慢的性能体验（大部分需要链接互联网）
- 用户体验较差*
- 支持图形和动画效果较差
- 不适用于应用商店及没有靠下载应用盈利机会
- 需要链接互联网
- 限制用户使用功能（比如，相机、GPS等）*

Web应用完全用HTML、JavaScript和CSS等Web技术开发，通过移动设备的浏览器来访问

随着HTML5的普及，上面提到的两个*的功能在Web应用中已得到很好的改进，尽管该技术性能提高了，但是依然无法与本地应用程序相媲美。

Web App就是运行于网络和标准浏览器上，基于网页技术开发实现特定功能的应用。

Hybrid App

Hybrid App兼具了Native App的所有优势，也兼具了Web App使用HTML5跨平台开发低成本的优势。

优点：

- 支持多平台访问
- 手机功能都可访问
- 适用于应用商店
- 部分支持离线功能

缺点：

- 未知的部署时间
- 用户体验不如本地应用
- 性能速度较慢（需链接网络）

	Web App（网页应用）	Hybrid App（混合应用）	Native App（原生应用）
开发成本	低	中	高
维护更新	简单	简单	复杂
体验	差	优	优
Store或market认可	不认可	认可	认可
安装	不需要	需要	需要
跨平台	优	优	差

Hybrid App详解

Hybrid App通常分为三种类型：**多View混合型，单View混合型，Web主体型。**

多View混合型

Native View和Web View独立展示，交替出现。 目前常见的Hybrid App是Native View与WebView交替的场景出现。

这种移动应用主体通常是Native App，Web技术只是起到补充作用。开发难度和Native App基本相当。

单View混合型

同一个View内，同时包括Native View和Web View。互相之间是覆盖（层叠）的关系。

这种Hybrid App的开发成本较高，开发难度较大，但是体验较好。如百度搜索为代表的单View混合型移动应用，既可以实现充分的灵活性，又能实现较好的用户体验。

Web主体型

移动应用的主体是Web View，主要以网页语言编写，穿插Native功能的Hybrid App开发类型。

这种类型开发的移动应用体验相对而言存在缺陷，但整体开发难度大幅降低，并且基本可以实现跨平台。Web主体型的移动应用用户体验的好坏，主要取决于底层中间件的交互与跨平台的能力。

（常见中间件appMobi、PhoneGap、AppCan）

	多View混合型	单View混合型	Web主体型
常见主体	Native	Native	Web
开发成本	中	高	低
用户体验	良	优	差

携程旅行app

携程旅行app，是一款深受用户喜爱的Hybrid App应用，它的开发模式，应该是Hybrid中多View混合型。

Native 页面

混合页面

Hybrid App的H5布局

HTML5 的页面被嵌入到 Native App 的 webView 中，此时是使用的设备的webKit内核。webKit内核中的一些私有的meta标签，这些meta标签在开发Hybrid App时起到非常重要的作用。

<meta content=" width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0;" name="viewport" />

强制让文档的宽度与设备的宽度保持1:1，并且文档最大的宽度比例是1.0，且不允许用户点击屏幕放大浏览；

<meta content="yes" name=" apple-mobile-web-app-capable" />

phone设备中的safari私有meta标签，它表示：允许全屏模式浏览；

<meta content="black" name=" apple-mobile-web-app-status-bar-style" />

iphone的私有标签，它指定的iphone中safari顶端的状态条的样式

<meta content="telephone=no" name=" format-detection" />

告诉设备忽略将页面中的数字识别为电话号码。

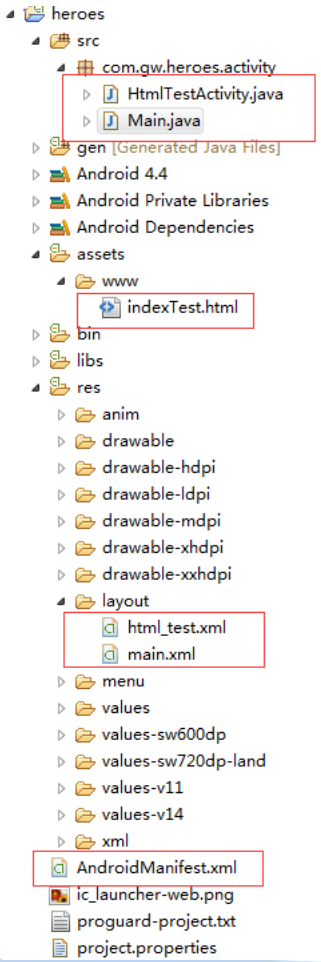
```
<meta charset="utf-8">
<link rel="dns-prefetch" href="http://res.m.ctrip.com">
<title>携程旅行-酒店预订,机票预订查询,旅游度假,商旅管理-携程无线官网</title>
<meta name="description" content="携程旅行网是中国领先的在线旅行服务公司，旗下携程旅行向超过9000万会员提供酒店预订、酒店点评及特价酒店查询、机票预订、假预订、商旅管理、为您的出行提供全方位旅行服务。">
<meta name="keywords" content="酒店预订,特价酒店,机票,机票预订,飞机票查询,航班查询,酒店团购,旅游度假,旅行,商旅管理,手机订酒店,手机订机票,携程旅行">
<meta name="viewport" content="width=320.1,initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0, user-scalable=no">
<meta name="apple-mobile-web-app-title" content="携程旅行">
<meta name="apple-mobile-web-app-capable" content="yes">
<meta content="telephone=no" name="format-detection">
<meta name="baidu-tc-cerfication" content="ba07d70ead40fe3171a72dc4099ed4e5">
<meta name="viewport" content="width=device-width,initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0, user-scalable=no, minimal-ui">
```

Viewport中，width=320,表示页面会默认自动缩放到跟手机屏幕相等的宽度，但有一个致命的缺点，对于高密度和超高密度的手机设备，页面（特别是图片）会失真，而且密度越多，失真越厉害。

width= device-dpi 手机设备就会按照真实的像素数目来渲染，用专业的话来说，就是1 CSS pixels = 1 device pixels。比如对于 640*960的 iphone，我们就可以做出 640*960的页面，在iphone上显示也不会有滚动条。这种方案可以在特定的分辨率下完美呈现，但是随着要兼容的不同分辨率越多，成本就越高，因为需要为每一种分辨率书写单独的代码。

width=320.1主要是为了解决iphone5，IOS6下的640*1136分辨率问题,页面顶部和顶部会出现一条黑色区域,当然不指定width也可以解决这个问题。

一个简单的Hybrid App



代码结构

交互前提

```
webview = (WebView) findViewById(R.id.webview);
WebSettings webSettings = webview.getSettings();
// 加上这句话才能使用javascript方法
webSettings.setUserAgentString(DESKTOP_USERAGENT);
String userAgent = webview.getSettings().getUserAgentString();
webSettings.setJavaScriptEnabled(true);
webview.requestFocus();
webSettings.setDomStorageEnabled(true);
webSettings.setJavaScriptCanOpenWindowsAutomatically(true);
webview.setWebChromeClient(new MyWebChromeClient());
webview.setWebViewClient(new MyWebViewClient());
webview.loadUrl("file:///android_asset/www/indexTest.html");
```

```
//设置可以支持缩放
mWebSettings.setSupportZoom(true);

//设置出现缩放工具
mWebSettings.setBuiltInZoomControls(true);

//设置默认缩放方式尺寸是far
mWebSettings.setDefaultZoom(WebSettings.ZoomDensity.FAR);

//设置允许访问文件数据
mWebSettings.setAllowFileAccess(true);

//设置是否保存密码
mWebSettings.setSavePassword(true);

//设置网页默认编码
Settings.setDefaultTextEncodingName(encoding);
```

```
class MyWebViewClient extends WebViewClient {

    // 重写shouldOverrideUrlLoading方法，使点击链接后不使用其他的浏览器打开。
    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
        view.loadUrl(url);
        // 如果不需要其他对点击链接事件的处理返回true，否则返回false
        return true;
    }

    @Override
    public void onPageFinished(WebView view, String url) {
        // 开始
        super.onPageFinished(view, url);
    }
}

class MyWebChromeClient extends WebChromeClient {

    // 设置网页加载的进度条
    @Override
    public void onProgressChanged(WebView view, int newProgress) {
        super.onProgressChanged(view, newProgress);
    }

}
```

Android Native调用JS中的方法

点击刷新html

```
Button button = (Button) findViewById(R.id.button); // 获取button控件
button.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {
        webview.loadUrl("javascript:updateHtml()");
    }
});
```

传入参数，改变页面图片

```
ContentResolver resolver = getContentResolver();
String path;
Uri originalUri = data.getData();
String[] proj = { MediaStore.Images.Media.DATA };
@SuppressWarnings("deprecation")
Cursor cursor = MediaStore.Images.Media.query(resolver,
        originalUri, proj);

if (cursor != null) {
    // 按我个人理解 这个是获得用户选择的图片的索引值
    int column_index = cursor
            .getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
    if (cursor.getCount() > 0) {
        cursor.moveToFirst();
        path = cursor.getString(column_index);
        webview.loadUrl("javascript:changePicURL('" + path + "')");
        System.out.println("----requestCode: ");
        cursor.close();
    }
}
```

JS调用Native中的方法

绑定一个对象到window，window.login

```
// 增加接口方法,让html页面调用 第二个参数为别名，在html 中调用会使用
HtmlTestJSObject jsObject = new HtmlTestJSObject();
webview.addJavascriptInterface(jsObject, "login");
```

Login对象包含的内容：

```
<div style="width: 100%;height: 15%">
    <a onclick="window.login.startCamera()" href="";>调用摄像头方法</a> <br />
    <a onclick="window.login.startChoosePhoto()" href="";>调用相册</a>
</div>
```

```
public class HtmlTestJSObject {

    @JavascriptInterface
    public void startCamera() {
        takePhoto(Math.random() * 1000 + 1 + ".jpg");

    }

    @JavascriptInterface
    public void startChoosePhoto() {
        choosePhoto();
    }
}
```

IOS Native调用JS的方法

本质就一个方法,通过 stringByEvaluatingJavaScriptFromString,都是同步。

- (NSString*)stringByEvaluatingJavaScriptFromString:(NSString *)script;

script 就是 JS 代码,返回结果为 js 执行结果。

```
function testFunction(abc){

  return abc;

};
```

webview 调用此 JS 代码如下:

```
NSString *js = @"testFunction('abc')";

NSString *result = [webView stringByEvaluatingJavaScriptFromStr
ing:js];
```

JS调用Native

通常方法：js修通过改doucument的loaction或者新建一个看不见的iFrame,修改它的 src,就会触发回调 webView 的 shouldStartLoadWithRequest,参数 request 的 url 就是新赋值的 location 或者 url,上层截获这个 url 的参数,对此分发即可。 这个都是异步调用的。

```
var messagingIframe;
  messagingIframe = document.createElement('iframe');
  messagingIframe.style.display = 'none';
  document.documentElement.appendChild(messagingIframe);

  function TestIOSJS(){
    messagingIframe.src = "ios/test/click";
  };
```

触发

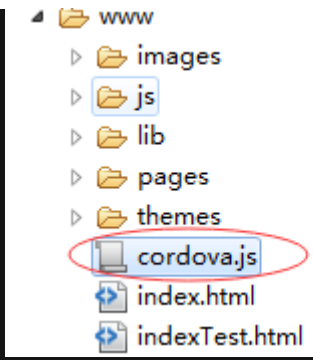
```
- (BOOL)webView:(UIWebView *)webView shouldStartLoadWithRequest:
(NSURLRequest *)request navigationType:(UIWebViewNavigationType)
navigationType
{
  NSString *url = request.URL.absoluteString;
  if([url hasSuffix:@"ios/test/click"]){
    //do something you want
    return NO;
  }
```

PhoneGap交互

JS中创建plugin

```
*/
var Update = function(){};

Update.prototype.openFile = function(fullPath,onSuccess, onFail){
    cordova.exec(onSuccess, onFail, 'update', 'openFile', [fullPath]);
};
Update.prototype.openApp = function(packageName,onSuccess, onFail){
    cordova.exec(onSuccess, onFail, 'update', 'openApp', [packageName]);
};
cordova.addConstructor(function() {
            if (!window.plugins) {
                window.plugins = {};
            }
            window.plugins.updatePlugin = new Update(); // addPlugin方法取消后用这种方式创建插件
        });
```



window.plugins.updatePlugin.openApp('URL',function(){},function(){});

配置文件中配置指令

```
104     <!-- Deprecated plugins element. Remove in 3.0 -->
105     <plugins>
106         <plugin name="share" value="com.gw.boutiquenews.plugin.SharePlugin"/>
107         <plugin name="storage" value="com.gw.boutiquenews.plugin.SharedPreferencesPlugin"/>
108         <plugin name="update" value="com.gw.boutiquenews.plugin.UpdatePlugin"/>
109         <plugin name="exitApp" value="com.gw.boutiquenews.plugin.ExitAppPlugin"/>
110     </plugins>
```

Java类中接收指令

```
28 public class UpdatePlugin extends CordovaPlugin {
29     @Override
30     public boolean  execute(String action, JSONArray args, CallbackContext callbackContext) throws JSONException {
31         PluginResult.Status status = PluginResult.Status.OK;
32         if (action.equals("openFile")) {
33             try {
34                 openFile(new File(args.getString(0).replaceAll("file://", "")));
35                 return true;
36             } catch(Exception e){
37                 PluginResult mPlugin = new PluginResult(PluginResult.Status.ERROR,"打开错误");
38                 mPlugin.setKeepCallback(true);
39                 callbackContext.sendPluginResult(mPlugin);
40                 return false;
41             }
42         }
43         else if(action.equals("openApp")){
44             try
45             {
46                 if(checkApp(args.getString(0))){
47                     openApp(args.getString(0));
48                 }
49             else{
50                 //Log.d("openApp", "没有应用");
51                 PluginResult mPlugin = new PluginResult(PluginResult.Status.ERROR,"没有应用");
```

自己如何做一个壳

- 1：做个app的icon
- 2：做个app的欢迎图
- 3：写一个webwiew即可

成熟的HTML5技术有

- Jquery Mobile前台界面
- Bootstrap后台、前台界面
- EasyUI后台界面
- Amaze UI前台界面
- Zepto.js、AgularJS、CanJs html5js框架
- Quintus/ canvasengine/ enchantjs等html5游戏引擎
- FastClick(点击事件类库，回头可以试试)
- Chart.js（图标类库）