JS&&Native通信

新的通信原理

JS调用Native

1.Android & iOS实现

iOS和Android在和JS通信方式上，有一些差异，需要各自按规范实现后，由js-sdk统一做兼容处理。Native的接口实现约束如下：

Native平台	JS调用Native 通信方式	Native接口实现

iOS	WKWebView中，OC通过WKScriptMessageHandler协议可以监听js中的
	
	window.webkit.messageHandlers.<name>.postMessage(<messageBody>)
	
	来响应JS调用
	
	参考：http://blog.csdn.net/u011619283/article/details/52135988
		▪ 实现 WKScriptMessage.
		ler" 的处理
		▪ 解析 WKScriptMessage.
		(即JS请求时传给Native
		
		{
		handler: "getUserInfo
		params: jsonObject,
		callback_name: "xxx"
		}
		
		1.执行Native业务方法，
		2.完成后，调用JS回调，

Android	Android可以在webview中，实现js的window.自定义对象.自定义方法，从而响应js调用，如下： mWebView.addJavascriptInterface(new JavaScriptObject(mContext), "someObjForJS"); public void someMethodForJS() {}	▪ 为JS创建 jsBridgeInAn ▪ 实现 jsBridgeInAndroi ▪ 接收如上面iOS一样的参

2.JS-SDK

墨客端内webview的H5业务，都需要引入此sdk，以实现对两端Native的兼容通信。@银超

文档

https://fe.ink.sohu.com/webview/view/docs/#/

H5 demo

https://fe.ink.sohu.com/webview/view/demo/

Native调用JS

直接执行JS代码串，如 fn(1,2)

安全限制

引入通信白名单机制，只有以下host下的H5页面，才能和Native通信，其他域下的通信，Native不做任何响应。

- *.sohu.com
- *.sohuno.com
- 10.*.*.* （IP段 内网使用，便于调试）
- 192.168.*.* （IP段 内网使用，便于调试）

JS调用Native规范约束

必读

JS对Native的调用分为三部分：

- 跳转页面
- 获取数据
- 调用功能（多数需要native回调，其载体也可能是页面，比如绑定手机号）

参数规范：

Handler	Params	responsData	responsData > status

驼峰式命名	▪ json object，不需要时传 null
	▪ key，命名统一使用 a_b 形式
	▪ key，不需要的key，默认不传，
	但是如果传了null，native也要可以处理
	▪ value，bool型统一标准 true|false
		▪ 任何handler都需要回调
		▪ json object，无返回数据时传 null
		▪ key，命名统一使用 a_b 形式
		▪ 有返回数据的统一遵循格式
		{
		status:200, // require int
		msg:"", // require string
		暂debug用，取值不做规范
		data:null // require jsonObject
		}
			语义遵循http状态码，如下：
			▪ 200 操作成功
			▪ 201 已操作成功过
			▪ 400 参数error
			▪ 500 操作失败
			▪ 501 暂未实现
			▪ 503 操作过程中被取消
			如无特殊说明，每一个业务接口都需要实现以

以下业务的responseData，均遵循上述规范，and，下面的文档中仅描述responseData.status 和 responseData.data。

JS调用Native业务列表

一、跳转页面

一些Native页面（包括新webview打开某些H5）跳转。使用端内URL Scheme跳转，参考文档：URL Scheme业务List

APP版本支持：

- iOS5.2+
- Android3.0+

备注：

iOS4.0 - 5.1（仅消息详情webview）以及Android2.1的历史实现如下：

~~Handler:~~

~~jumpToPage~~

~~Params:~~

~~{~~

"destinationPage":"FeedPage|MessageCenter",//android2.1 && ios5.1-
消息中心webview使用(意见反馈页|消息中心页)，日后新版本不再维护 }

注：

url参数 ios5.1+,android3.0+开始支持

responseData：

无

二、获取数据

1.获取用户信息

Handler：

getUserInfo

Params：

null

responseData.status ：

需实现所有status

responseData.data ：

```
{
    "nickname": "ink00001112", //require
    "user_id": "48c63a82-7d74-11e6-b5ba-968e5335f5e6", //require
    "message_id": 2, //弃用。历史遗留问题
消息详情页，ios因为url中没有附加message_id参数，所以需要此接口返回message_id；Android不需要；ios
5.2后去掉此参数，message_id需要拼接到server下发的URL里，最好是server追加上
    "token": "68c28c2770becc8328e9d5117e098908ad478d2b", //require
    "device_id": "BC47ACA7-1CAA-4949-B1B7-31E664BD5D2C", //require
    "has_bind_phone": true, //require
    "headers":jsonObject  // native端所有header，参见文档
http://wiki.sohu-inc.com/pages/viewpage.action?pageId=22315055
}
```

APP版本支持：

- iOS5.2+
- Android2.1+

备注：

把标记用户、设备的信息都归集到此接口，完全merge ios4.0-5.1的getHttpHeader接口。

2.获取HttpHeader

获取H5向server请求所需要的httpHeader。

Handler：

getHttpHeader

Params：

null

responseData：

```
{ "APP-VERSION":"4.0.1", "AUTHORIZATION":"16244d4053dc5d6ccfffaff443d745bf9a0243b0",
"DEVICE-ID":"BC47ACA7-1CAA-4949-B1B7-31E664BD5D2C", "UID":"92dc39ce-8efc-11e7-88b3-66d6c09cc091" }
```

APP版本支持：

- iOS4.0+（仅消息详情webview）
- iOS5.2+

备注：

日后计划停止维护此接口，增量字段都放到getUserInfo中

三、调用功能

1.绑定手机

Handler：

bindPhone

Params：

null

responseData.status ：

需实现所有status

responseData.data：

null

APP版本支持：

- iOS5.2+
- Android3.0+

2.分享

h5自定义分享渠道以及分享内容，暂只支持分享h5URL，图片等资源的分享，日后再说。

关于分享成功或者失败的Toast提示显示时机问题，请遵循以下原则：

- 只有当十分确定分享成功或者失败时给Toast，不确定时不给Toast（比如分享完了留在了微信qq，后面他再回来墨客，native是无法知道成功还是失败的）

Handler：

shareInfo share

Params：

```
{
    channel_list: ["wechatMessage", "wechatTimeline", "QQ", "weibo", "copyLink"], //require.
需要显示的分享渠道白名单 数组标记顺序 下面map必须要有对应
    "wechatMessage":
    {
  media_type:"h5", //require 日后可能支持更多，如img|video
        title: "标题",//require string
  content:'内容', // require string
  description:"",//require string
        thumb_url: "", //option string 默认为墨客icon
        url: "" //require string 资源URL 暂只支持h5URL
    },
    "wechatTimeline":
    {
  media_type:"h5",
        title: "标题",
        content:'内容',
  description:"",
        thumb_url: "",
        url: ""
    },
    "QQ":
    {
  media_type:"h5",
        title: "标题",
        content:'内容',
  description:"",
        thumb_url: "",
        url: ""
    },
    "weibo":
    {
  media_type:"h5",
        title: "标题",
        content:'内容',
  description:"",
        thumb_url: "",
        url: ""
    },
    "copyLink":
    {
        url: ""
    }
}
```

responseData.status ：

需实现所有status

responseData.data：

```
{
 channel:"wechatMessage"
}
```

key	description	value
channel	分享渠道	none | weibo | wechatMessage | wechatTimeline | ~~huyou~~ | copyLink | QQ | ~~QZone~~

APP版本支持：

- iOS5.2+
- Android3.0+

备注：

1. 用户分享成功后，在第三方APP未选择返回墨客的，可以忽略

3.设置webview 选项菜单

webview标题栏右上角的菜单按钮。（暂默认右上角无按钮，以后优化）

本接口设计初衷是，支持JS配置icon、红点、下拉菜单、点击交互。（红点在图标左上角，下拉菜单项文字不超过4个字等@产品？，下拉项不超过4个？）

菜单icon及下拉菜单项的触感反馈遵守Native规范。

Handler：

optionMenu

Params：

```
{
    "icon": "http://x.jpg", //require。菜单图标URL
    "redpoint": 1, //option int，默认0。是否显示红点
    "calljs": "js code", //option。点击native图标时的js code回调
    sub_menus: [ //option。下拉菜单列表
    {
        "icon": "http://x.jpg", //option
        "redpoint": 0, //option
        "text": "消息", //require
        "calljs": "js code" //require
    },
    {
        "icon": "http://x.jpg",
        "redpoint": 1,
        "text": "分享",
        "calljs": "js code"
    }]
}
```

- params为 null 时，Native隐藏右上角菜单按钮

responseData.status ：

需实现所有status，除 201 503

responseData.data：

null

APP版本支持：

- iOS5.2+
- Android3.0+

4.准备修正H5中的键盘弹出

ios先保留此接口，Android视后续webview内兼容表现情况，再决定是否需要Native给H5提供input等替代方案。

Handler：

setIsInputUnderSoftKeyboard

Params：

```
{
    "isInputUnderSoftKeyboard": true/false
//Native在键盘将要弹出的时候，是否强制将网页内容上拉一个键盘高度
}
```

responseData.status ：

需实现所有status，除 201 503

responseData.data：

null

APP版本支持：

- ~~iOS4.0 - 5.1（仅club webview）~~
- iOS5.2+

5.切换离线代理状态

用来做native对h5的URL本地化代理。因为iOS系统限制，代理的实现依赖开启NSProtocol，但是开启的时候系统会默认截掉post请求的body体，所以，对iOS来说，需要H5主动调用native接口，控制
打开和关闭状态。（相关离线化实现方案，参见：端内H5离线加载及缓存更新方案）

Liu XueFeng(社交产品中心) 记得看一下这个↓

*为了统一离线代理开关设置，减少不必要的匹配耗时，Android也实现了此协议。请js业务端不需要代理时记得关闭以提高全局请求速度。

注意：

- native：Webview初始化的时候，需要默认打开。初始化之后第一个loadUrl请求，如果不符合h5AppCache config.json中的规则，那么关掉NSProtocol。（避免其他三方网页无法正常使用）
- JS：需要在页面中的资源都加载成功之后，关掉NSProtocol，以保证后续的post请求，可以正常访问。~~（或者每次post开始前都手动关闭，然后response结束后，手动开启）~~

Handler：

switchLocalProxy

Params：

```
{
    "is_open": true/false //默认为true
}
```

responseData.status ：

需实现200 400 500.

responseData.data：

null

APP版本支持：

- iOS5.2.1+
- Android 3.1+

Native调用JS业务列表

尽量避免这类的直接调用。暂无。