# JSBridge文档

限于时间，本文档仅介绍H5、Native双方需要遵守的规范，维护API的主要注意事项，以及调用范例。

## API命名规范：

Native端 ：NF\_functionname

e.g. NF\_OPENGPS 打开GPS并返回位置信息

说明：NF means native function ，functionname按照业务需求进行命名，再如打开相机:NF\_OPENCAMERA

H5端 ：JF\_functionname

e.g. JF\_LOGIN 执行H5内的登录逻辑（此处我难以找到合适的例子，仅用一个来说明命名规范）

JF means javascript function，同样 functionname按照业务需求进行命名。

## 数据规范

从消息通信底层而言，所有的消息都按照规定的格式封装于JSON中并进行了序列化，但是这并不是API业务层需要过多关心的，两端数据交互都是按照json格式规范，具体业务的 数据结构体 会在业务需求制定后定下来，而key的命名按照小驼峰法。

## API实现、注册及维护

native端，此文中略

H5端，对于js\_API的设计框架，我并不在行，请H5相关同事费心，要注册一个js\_api供native调用需要满足以下条件：

1. H5JSBridge\_V1\_0.js文件已经加载
2. 加载相关的API的js文件

例如：

|  |
| --- |
| <script type=*"application/javascript"* charset=*"utf-8"*  src=*"JSAPI\_name.js"*></script> |

*JSAPI\_name.js*的内容：

|  |
| --- |
| connectWebViewJavascriptBridge(**function**(bridge) {  bridge  .registerHandler(  "test2",  **function**(data, responseCallback) {  //data 是native 发送的数据，此处实现该JsAPI业务，得到responseData 通过回调  //将处理结果返回Native 注意定义好Bean    //此处打log看一下数据是否收到，非必须  log("data from Native:",data)    //此处实现API逻辑  //...    //此处返回处理结果给native  **var** responseData = {"key1":"i am result form js"};  responseCallback(responseData);  });  }) |

此处注意：test2即为注册的API名，按照规范，可能我们需要改为JF\_TEST。

我建议将每个API写到单个JS文件，便于维护。

## 调用

在API已经注册的前提下，调用API按照如下形式：

### Native端（以Android为例）

|  |
| --- |
| webView.callHandler("test2", "data from Java",  **new** CallBackFunction() {  @Override  **public** **void** onCallBack(String data) {  // **TODO** Auto-generated method stub  Log.*i*(TAG, "reponse data from js " + data);  }  }); |

函数原型：[网页载体对象].callHandler(String JS\_APIName , String data2JS , CallBackFunction callback)

* JS\_APIName: js注册的API的名称；
* data2JS: 发送给JS的数据，按照约定，应该是JSON数据，经过序列化并且是String对象；
* callback：回调接口对象，当js\_api业务逻辑完成后，会回传结果，onCallBack（String data）中实现Native的后续逻辑

例如以下情景：Native存在功能：将当前的web页面用外部浏览器打开，从实际来看我们不需要向JS发送额外的数据，假定注册的API名为JF\_GETURL

|  |
| --- |
| webView.callHandler("JF\_GETURL", "",  **new** CallBackFunction() {  @Override  **public** **void** onCallBack(String data) {  // **TODO** Auto-generated method stub  Log.*i*(TAG, "reponse data from js " + data);  //解析data 获得url  //调用外部浏览器打开该url  }  }); |

### H5端

以打开gps为例：

|  |
| --- |
| **function** callNativeOpenGPSAndResponseLocation(onNativeResponseCallback) {  connectWebViewJavascriptBridge(**function**(bridge) {  bridge.callHandler('NF\_OPENGPS', {  'key1' : 'this is test data,this native api do not need data'  }, onNativeResponseCallback)  })  } |

可以看到，我们已经进行了一定的封装，这样可以避免频繁输入约定的API名而出错（毕竟IDE还没法给我们建议这些内容），同样，在Native端做这样的工作也是合情合理的。

实质上调用GPS并不需要额外的数据，即使传了也不会影响（不要故意弄个错的格式来调皮，会抛异常）

onNativeResponseCallback是一个function，含义NativeAPI处理完成后回传数据JS的回调事件，例如：

|  |
| --- |
| **function** testGPS() {  callNativeOpenGPSAndResponseLocation(**function**(responseData) {  log("response form native", responseData);  });  } |