
上海邮乐


Webgum 项目

设计文档



文档版本：V1.0.2

文档日期：2018 年 2 月 8 日

	文档名称：webgum 项目设计文档	版本： 1.0.2
	文档编号：	日期: 2018-03-06

版权声明


本文件中出现的任何文字叙述、文档格式、插图、照片、方法、过程等内容，除另有特别注明，版权均属上海邮乐网络技术有限公司所有，受到有关产权及版权法保护。任何个人、机构未经上海邮乐网络技术有限公司的书面授权许可，不得复制或引用本文件的任何片断，无论通过电子形式、非电子形式或其他形式。

【摘要】

关键词：Webgum，H5，Android，Object-C，C/C++，react-native


版本记录

版本编号	版本日期	修改者	说 明
V1.0.1	2018-01-16	李岩	创建
V1.0.2	2018-02-08	李岩	1. 添加标准及规范

	文档名称：webgum 项目设计文档	版本： 1.0.2
	文档编号：	日期: 2018-03-06

目 录

1. 背景.....	1
2. 要求及可行性分析.....	1
3. 设计原理.....	3
4. WEBGUM 标准及规范.....	7

	文档名称：webgum 项目设计文档	版本： 1.0.2
	文档编号：	日期: 2018-03-06

1. 背景


目前邮乐系项目有很多，它们大多运行在 Android、IOS、PC 等系统平台上。随着业务及项目的发展，这些原生项目与 H5 之间的交互越来越紧密；同时，H5 在获取诸如设备、硬件等原生 API 信息时，也迫切的需要原生项目的支持。

由于邮乐项目众多，又涉及到多个系统平台，各项目与 H5 之间的交互及规则制定又都处在各自为政的状态。这使得 H5 在与原生项目交互时存在诸多弊端，比如：H5 在获取同一原生信息（如：地理位置）时，在不同项目不同系统中需要调用不同的方法；在多个项目中引入同一个 H5 时，那么该 H5 页面必须要做到同时具备多套与原生交互的方式以确保在每个项目中都能正常运行；当一个 H5 页面要被嵌入到一个新的项目时，必须要经过二次开发以完成其与新原生项目的交互等。这无疑加重了前端开发人员的工作，存在大量的重复代码，也违背了 H5 跨系统跨平台的初衷。

为了达到 H5 不经过二次开发就能够跨平台跨项目正常运行的目的；增强其可移植性、可复用性等优势，就必须打破项目与系统平台之间的壁垒，统一 H5 与各邮乐项目的交互规则。Webgum 应运而生。它作为 H5 与原生交互的中转站，可以提供统一的 API 供上下游双方调用。

2. 要求及可行性分析

鉴于 Webgum 需要完成消除项目及平台对 H5 的影响，以及方便对其开发、维护和使用，因此它必须具备以下几个特点：

	文档名称：webgum 项目设计文档	版本： 1.0.2
	文档编号：	日期: 2018-03-06

1) 通用性

Webgum 本就是为了解决 H5 与各项目个平台交互的问题而产生的 ,所以它也必须可以在常用系统平台及各项目中正常运行。

2) 兼容性

在现有的很多项目中已经存在 H5 与原生的交互 , Webgum 的引入不应破坏已有的规则。只是增加一种标准的交互方式 , 既做到与已有规则的兼容。

3) 易用性

使用 Webgum , 必须让 H5 与原生的交互更加简单和易用。

4) 标准性

Webgum 是 H5 与原生交互的中间件供双方调用。更应是一种统一的调用标准。只要遵守这种标准 , 就能让 H5 不必关心其运行环境 , 从而完成 H5 对平台或项目的解耦。


5) 独立性

Webgum 不应该影响或依赖宿主项目的其他功能 , 可以被其宿主项目任意的引入或删除。

6) 扩展性

由于 H5 调用不同原生项目时需要的信息各式各样 , 侧重点也不尽相同。所有 Webgum 不应是一个整体 , 而应是模块化插件化的组装 , 模块都必须遵守 Webgum 的标准 , 各模块之间相互独立 , 互不影响。

目前 , 在各项目个系统平台上 , H5 与原生的交互均已有成熟的实现方式 , 而 Webgum

	文档名称：webgum 项目设计文档	版本： 1.0.2
	文档编号：	日期: 2018-03-06

只是对不同实现方式做规范化。因此，Webgum 项目在技术上是可行的。

3. 设计原理

在 js 与各语言交互中，越上层的语言局限性越大，C++与 js 交互是最自由限制性最小的。而 Android/IOS 与 js 的交互就有一定的局限性了，比如 js 在调用原生的时候参数不能传递 js 对象，只能传递基础数据类型，原生的返回值也只能是基础数据类型和一些特定对象，不能传递诸如 list、set 或 map 这些高级数据对象。Rn 与 js 的交互局限性就更强了，参数和返回值都只能是基础数据类型，而且 js 调用 rn 的返回值也必须通过回调的方式获取。

两种思路：


1) 借鉴 react-native 或 cordova 的方式

react-native 和 cordova 都是通过请求回调的方式来完成交互。比如在 react-native 中，H5 通过调用 window.postMessage(data)的方法来调用原生，原生可以根据 data 的不同来完成对请求的处理，并回调 H5 的方法。在 github 上面，有很多类项目使用类似的方法来完成交互，如 jsBridge 等。

2) 采用注入对象的方式

对于 C++、Android 或者 ios，都可以注入一个 js 的全局对象，通过这个对象可以完成交互。

以上两种方式是最普遍也是最常用的 H5 与原生交互的方式，但是两者都各有利弊。第一种方式可以更简单的实现控制层与业务逻辑层或数据层的分离，更易实现模块化，更易

	文档名称：webgum 项目设计文档	版本： 1.0.2
	文档编号：	日期: 2018-03-06

扩展，自由化程度较高，对于耗时操作或对于某一事件的持续监听在没有回调之前不会影响 H5，既与 H5 进程或线程是异步的。但是需要自己实现控制层，对于 H5 来说这种方式往往使项目书写难度加大，某一功能的代码相对离散，比较难维护，而且对于即时性的或者不耗时的交互，这种方式也显得比较笨重。第二种方式不需要自己实现控制层，对于 H5 来说更易书写，某一功能的代码相对聚合，即时性或者不耗时的交互书写轻便。但是对于系统的依赖较强，有一定的局限性，对于耗时操作或对于某一事件的持续监听会影响 H5，最重要的是这种方式对于 H5 与 m 的交互适用性很差。

综上所述，可以将两种方式结合，最大限度的发挥各自的优点。

Webgum 会在客户端使用 webView 时，在原有 userAgent 的末尾按照标准格式追加 Webgum 标识，并会注入一个 js 全局对象。H5 页面可以据此判断是否支持 Webgum，并通过 js 对象实现与原生的交互。

客户端的基本工作流程如下：


1) 加入 Webgum 标识

在使用时必须添加 Webgum 标识。标识格式为：

Webgum/Version (OS; ProjectName)

其中：

- a. **Version** 标识客户端的 Webgum 版本
- b. **OS** 表示客户端的运行环境

	文档名称：webgum 项目设计文档	版本： 1.0.2
	文档编号：	日期: 2018-03-06

- c. **ProjectName** 表示项目编号
- d. Version 及 OS 为必填项，ProjectName 为选填项，当存在 ProjectName 时，OS 与 ProjectName 用分号 ‘;’ 加空格 ‘ ’ 隔开
- e. 在 Version 与左括号 ‘(’ 之间用一个空格隔开
- f. 设置 Version、OS 及 ProjectName 时禁止使用中文
- g. 在整个标识中所出现的反斜线 ‘\’、小括号 ‘()’ 及分号 ‘;’ 均为英文输入法环境下的符号。

示例：

在 Android 的项目中可以在 userAgent 后追加如下：“Webgum/1.0.256 (Android)”；在 IOS 项目中可以追加如下：“Webgum/1.1.457 (IOS)”；在邮掌柜 App(IOS)react-native 中使用可以追加如下：‘Webgum/1.1.457 (RN-IOS; Yzg-IOS)’ 等。

2) 注入 js 对象

注入的 js 对象必须包含 Webgum 标识中对应版本的所有方法。通过该对象可以获取 Version，OS，ProjectName，插件列表等。不同客户端注入 js 对象的方法不一。


3) 插件管理

H5 的基本工作流程如下：

1) 引入 wg.js

在 wg.js 中会根据 userAgent 引入客户端注入的对象 封装成 wg 对象添加到 window 中。

2) 通过 wg.xxxx()与原生交互

	文档名称：webgum 项目设计文档	版本： 1.0.2
	文档编号：	日期: 2018-03-06

示例：可以通过 `wg.getVersion()` 获取客户端的 webgum 版本；通过 `wg.getOS()` 获取客户端的运行环境；通过 `wg.exit()` 关闭网页；通过 `wg.getPlugins()` 获取客户端插件名称列表

3) 通过 `wg.addListener()` 与原生交互

该方法主要用来实现对耗时操作的调用或对某一动作的监听。

4) 通过 `wg.getPlugin(plugin_name)` 获取模块对象，并通过模块对象与模块交互

在调用 `wg.getPlugins()` 之后会返回一个插件名称的列表(例如 `['battery' , 'camera']`)，将列表中返回的某一个插件名作为参数通过 `wg.getPlugin(plugin_name)` 获取一个插件，通过插件对象提供的方法与原生完成交互。

示例 1：

```
var version = wg.getVersion()
alert(version)

var battery = wg.getPlugin("battery")
var power = battery.getPower()
alert(power)
```


示例 2：

```
wg.addListener("network", "onNetChange", {"":""}, onChange)

function onChange(value) {

}

wg.addListener("location", "localCollect", {"timeInterval": "5000"}, onCollect)
```

	文档名称：webgum 项目设计文档	版本： 1.0.2
	文档编号：	日期: 2018-03-06

```
function onCollect(value) {
}
```

4. Webgum 标准及规范

webgum 在开发或扩展插件时，应遵循以下规范：

1) 基本规范

webgum 必须是一个主体和多个插件的组合，不应该存在多个主体，必须包含一个全局的 js 对象 wg（小写），作为整个项目的主体和运行的起点，其必须包含最基本的如获取基本信息，插件信息等功能。而插件则应是寄附在主体之上的功能模块，通过主体完成与 H5 的交互，各插件应各自独立，不应相互依赖。

2) 标识规范

参见设计原理中所述

3) 命名规范

在提供给 H5 的模块名、方法名、参数名以及返回数据应尽量采用小驼峰命名法。

4) 交互规范

必须提供**即时交互**和**回调交互**两种方式。


调用**即时交互**可分为有参和无参两种，对于参数个数不做限制，大致有以下几种：

wg.functionName()

wg.functionName(param)

wg.functionName(param1 , param2 , param3.....paramN)

这类方法必须有返回值，返回值应是一个 json 对象，在 json 对象中除了包含业务功能

	文档名称：webgum 项目设计文档	版本： 1.0.2
	文档编号：	日期: 2018-03-06

相关的数据外，还必须包含 code、msg、和 status 三个属性。其说明及数据类型如下：

参数	类型	说明
code	String	响应码
msg	String	响应信息
status	Boolean	响应状态

表 4.1：返回数据格式基本要求

在使用时可以通过 status 和 code 判断响应操作是否成功。

调用**回调交互**必须有至少一个回调函数作为参数，对于其他参数不做特殊要求，对于参数个数不做限制，大致有以下几种：

wg.functionName(callback)


wg.functionName(callback1 , callback2 , callback3.....callbackN)

wg.functionName(callback , param)

wg.functionName(callback1 , callback2 , callback3.....callbackN , param1 , param2 , param3.....paramN)

这类方法本身没有返回值，而是调用回调函数，回调可以有效一次或多次，视具体方法而定。在原生项目处理完成后会调用回调函数，并传递参数，参数类型及个数视具体方法而定，当参数类型为 js 对象时，可以提供 code、msg 和 status 等基本属性。也可以不提供。

另：对于普通参数和回调参数在先后顺序上应根据具体的方法来确定，可以相互穿插。

	文档名称：webgum 项目设计文档	版本： 1.0.2
	文档编号：	日期: 2018-03-06

5) 插件规范

每一个插件都应该提供一个获取插件基本信息的方法，用以提供插件名插件版本等信息。插件对象需要通过 wg 对象来获取，不应孤立存在。插件的命名规范也应遵守小驼峰命名法，插件也应支持即时交互和回调交互两种方式。大致方式如下：

```

wg.getPlugin(pluginName).functionName()
wg.getPlugin(pluginName).functionName(param)
wg.getPlugin(pluginName).functionName(param1 , param2 , param3.....paramN)
wg.getPlugin(pluginName).functionName(callback)
wg.getPlugin(pluginName).functionName(callback1 ,
    callback2,callback3.....callbackN)
wg.getPlugin(pluginName).functionName(callback , param)
wg.getPlugin(pluginName).functionName(callback1 , callback2 ,
callback3.....callbackN , param1 , param2 , param3.....paramN)

```

同样，即时交互的方法必须有返回值，回调交互的方法没有返回值，通过回调获取结果。与主体交互方式一致。

6) 版本规范

通常，为了便于管理，webgum 具有一个主版本号，各插件也应有各自的版本。主版本号只用来管理 webgum 项目的主体功能，与插件无关。而插件版本则应由插件来管理，其应标识出该插件对应某个主版本号或更高版本号的 webgum 上可用。

注：具体的标准及规范，可参阅公共 API 及各开发手册等文档。