



蚂蚁金服金融科技产品手册

H5 容器和离线包

产品版本：V20200421

文档版本：V20200421

蚂蚁金服金融科技文档

蚂蚁金服金融科技版权所有 © 2020，并保留一切权利。

未经蚂蚁金服金融科技事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

商标声明



及其他蚂蚁金服金融科技服务相关的商标均为蚂蚁金服金融科技所有。
本文档涉及的第三方的注册商标，依法由权利人所有。

免责声明

由于产品版本升级、调整或其他原因，本文档内容有可能变更。蚂蚁金服金融科技保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在蚂蚁金服金融科技授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过蚂蚁金服金融科技授权渠道下载、获取最新版的用户文档。如因文档使用不当造成的直接或间接损失，本公司不承担任何责任。

目录

1 简介	1
1.1 H5 容器简介	1
1.2 离线包简介	2
2 接入 Android	5
2.1 添加 Android SDK	5
2.2 使用 UC 内核	6
2.2.1 添加 UC SDK	6
2.2.2 扩展 UC 内核	7
2.3 使用容器	7
2.3.1 升级说明	7
2.3.2 使用 SDK	8
2.4 使用离线包	11
2.4.1 管理离线包	11
2.5 自定义 JSAPI	15
2.6 自定义功能使用说明	19
2.6.1 自定义导航栏 (10.1.68(beta))	19
2.6.2 自定义标题栏 (10.1.60)	27
2.6.3 H5 容器配置	43
2.6.4 H5 容器扩展	45
2.6.5 管理 H5 页面	48
2.7 使用教程	50
2.7.1 总览	50
2.7.2 在 Android Studio 创建应用	51
2.7.3 在 mPaaS 控制台创建应用	64
2.7.4 接入 mPaaS Inside 到工程	66
2.7.5 添加 H5 组件到工程	68
2.7.6 使用 H5 容器	69
2.7.7 使用 H5 离线包	84
2.8 Android 常见问题	96
3 接入 iOS	96
3.1 添加 iOS SDK	96
3.2 使用容器	97
3.2.1 10.1.60 升级指南	97
3.2.2 使用 SDK (版本 ≥ 10.1.60)	99
3.2.3 使用 SDK (版本 = 10.1.32)	107
3.3 使用离线包	111
3.3.1 管理离线包	111
3.4 自定义 JSAPI	114
3.5 自定义插件	121
3.6 使用教程	126
3.6.1 总览	126
3.6.2 在 mPaaS 控制台创建应用并下载配置文件	127
3.6.3 在 Xcode 创建工程	128
3.6.4 使用 H5 容器	129
3.6.5 使用 H5 离线包	143
3.7 参考说明	148
3.7.1 定制 H5 页面导航栏	148
3.7.2 H5 容器自动化埋点	155
3.8 iOS 常见问题	158
4 内置 JSAPI	176
4.1 启动参数	176
4.2 事件扩展	178
4.2.1 初始化完毕	178
4.2.2 点击标题栏	179
4.2.3 点击副标题栏	179
4.2.4 页面压后台	180

4.2.5 页面恢复运行	181
4.2.6 点击右上角按钮	183
4.2.7 回退	184
4.2.8 添加通知	185
4.2.9 移除通知	187
4.2.10 分发消息	189
4.3 页面上下文	191
4.3.1 打开新页面	191
4.3.2 关闭当前页面	194
4.3.3 关闭多个页面	196
4.3.4 启动其他应用	199
4.3.5 退出当前应用	202
4.3.6 startH5App	204
4.4 Native 功能	205
4.4.1 扫码解析	205
4.5 界面	207
4.5.1 警告框	207
4.5.2 确认框	209
4.5.3 弱提示	210
4.5.4 选择列表	212
4.5.5 设置标题	214
4.5.6 设置导航栏底部细线颜色	216
4.5.7 设置导航栏背景色	217
4.5.8 设置右上角按钮	218
4.5.9 显示右上角按钮	221
4.5.10 隐藏右上角按钮	222
4.5.11 显示加载中	222
4.5.12 隐藏加载中	224
4.5.13 显示标题栏加载中	225
4.5.14 隐藏标题栏加载中	225
4.6 工具类	226
4.6.1 获取容器的启动参数	226
4.6.2 截屏	228
4.6.3 RPC 调用	229
4.6.4 上报埋点	234
4.6.5 设置 AP 数据	236
4.6.6 获取 AP 数据	239
4.6.7 移除 AP 数据	241

1 简介

1.1 H5 容器简介

H5 容器是一款移动端 Hybrid 解决方案 SDK (Nebula SDK) , 提供了良好的外部扩展功能, 拥有功能插件化、事件机制、JSAPI 定制和 H5App 推送更新管理能力。

功能

H5 容器组件为您提供以下功能：

- 加载 H5 页面, 并按照会话 (Session) 的概念管理各个页面。
- 提供丰富的内置 JSAPI, 实现譬如页面 push 或 pop、标题设置等功能。
- 支持用户自定义 JSAPI 和插件功能, 扩展业务需求。
- 接入 H5App 后台, 方便管理离线或者在线 H5App, 实现类似 mPaaS 应用首页上应用的推送、更新。
- 自带埋点功能, 接入 H5 监控平台, 能够实时看到页面加载的性能、错误报警和监控。
- Android 使用 UCWebView, 拥有解决系统级 WebView Crash 的能力, 内存管理更合理, 网络加载提升更快, 兼容性更好。彻底告别了在 Android 下兼容不同 WebView 的问题。
- 支持自定义网络库, 自定义网络通道。

特点

H5 容器在以下方面有着出众而强大的特点。

稳定性

- 经过亿级用户考验, 崩溃率、ANR 率 (Application Not Responding , 应用程序无响应) 以及其他稳定性指标有保障。
- Android 平台基于 UCWebView 深度定制, 崩溃率和 ANR 率远低于系统 WebView, 拥有解决系统 WebView 问题的能力。

离线包能力

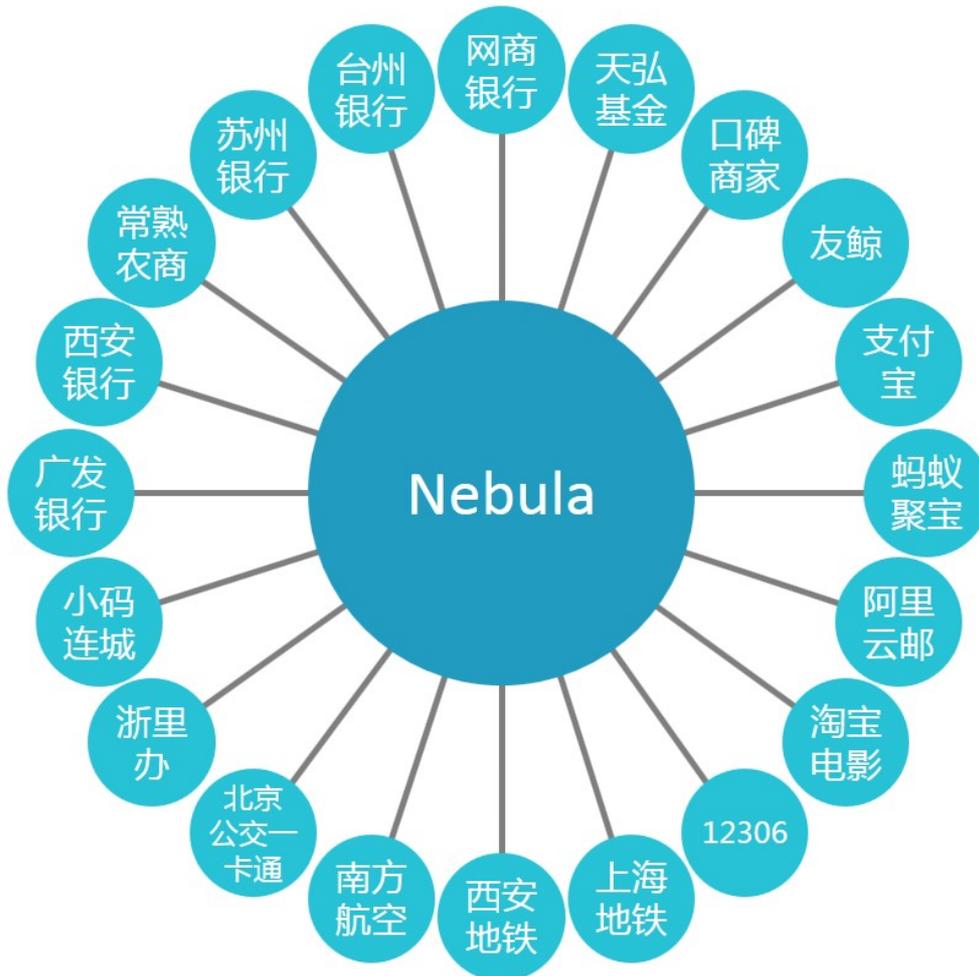
- **强大的离线包统一推包平台**：通过推包平台, 能够迅速将离线包推送到客户端, 保证客户端数据在短时间能够获得最新同步。
- **客户端多种策略, 增加离线率和更新率**：强制更新, 尝试更新, 强制离线, 尝试离线满足不同业务方需求。

客户端能力

- 自定义网络库, 客户端网络防劫持, 过滤小广告：
 - 支持 HTTP 2.0 网络请求, 多路复用请求。
 - WebView 的请求全部走 Native 通道, SPDY、HTTP 2.0 等各种网络协议用户自定义, 有效解决网络劫持问题、页面广告。支持 HttpDNS, 防止本地 DNS 解析失败。
- 自定义 JSAPI 插件, 可快速实现 JSAPI。

生态基础

接入了全部的蚂蚁 App，有稳定的生态基础。



1.2 离线包简介

传统的 H5 技术容易受到网络环境影响，因而降低 H5 页面的性能。通过使用离线包，您可以解决该问题，同时保留 H5 的优点。

离线包 是将包括 HTML、Javascript、CSS 等页面内静态资源打包到一个压缩包内。您预先下载该离线包到本地，然后通过客户端打开，直接从本地加载离线包，从而最大程度地摆脱网络环境对 H5 页面的影响。

使用 H5 离线包可以给您带来以下优势：

- **提升用户体验**：通过离线包的方式把页面内静态资源嵌入到应用中并发布，当用户第一次开启应用的时候，就无需依赖网络环境下载该资源，而是马上开始使用该应用。
- **实现动态更新**：在推出新版本或是紧急发布的时候，您可以把修改的资源放入离线包，通过更新配置让应用自动下载更新。因此，您无需通过应用商店审核，就能让用户及早接收更新。

离线包原理

您将从以下方面了解离线包原理：

- 离线包结构
- 离线包类型
- 渲染过程

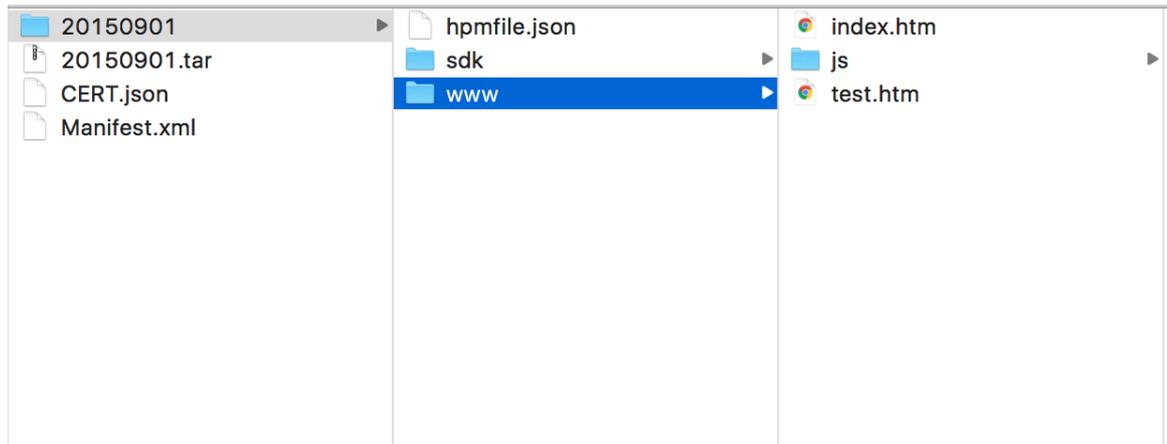
离线包结构

离线包是一个 .amr 格式的压缩文件，将后缀 amr 改成 zip 解压缩后，可以看到其中包含了 HTML 资源和 Javascript 代码等。待 H5 容器加载后，这些资源和代码能在 WebView 内渲染。

以 iOS 系统为例，下图显示了一般资源包的目录结构：

- 一级目录：一般资源包的 ID，如 20150901。

二级目录及往后即为业务自定义的资源文件。建议所有的前端文件最好保存在一个统一的目录下，目录名可自定义，如 /www，并设定当前离线包默认打开的主入口文件，如 /www/index.html。



离线包类型

通常，在 H5 的开发过程中，会存在使用一些基础通用库的情况，比如 zepto，fastclick 等。在 App 中的 WebView，有时候 CDN 的机制不可靠，曾经发现有机型在退出后，缓存自动失效。

为了进一步提升 H5 页面性能，使用全局离线包，将一系列的通用资源打成一个特殊的 App 包，下发到客户端。

离线包可以分为以下类型：

- **全局离线包**：包含公共的资源，可供多个应用共同使用。
- **私有离线包**：只可以被某个应用单独使用。

使用全局离线包后，在访问 H5 的时候，都会尝试在这个包尝试读取。如果该离线包里有对应资源的时候，直接从该离线包里取，而不通过网络。因此，全局离线包的机制主要是为了解决对于通用库的使用。

由于要保证离线包的客户端覆盖率以及足够的通用性，此包一般的更新周期至少为 1 个月，并且严格控制离线

包的大小。

渲染过程

当 H5 容器发出资源请求时，其访问本地资源或线上资源所使用的 URL 是一致的。

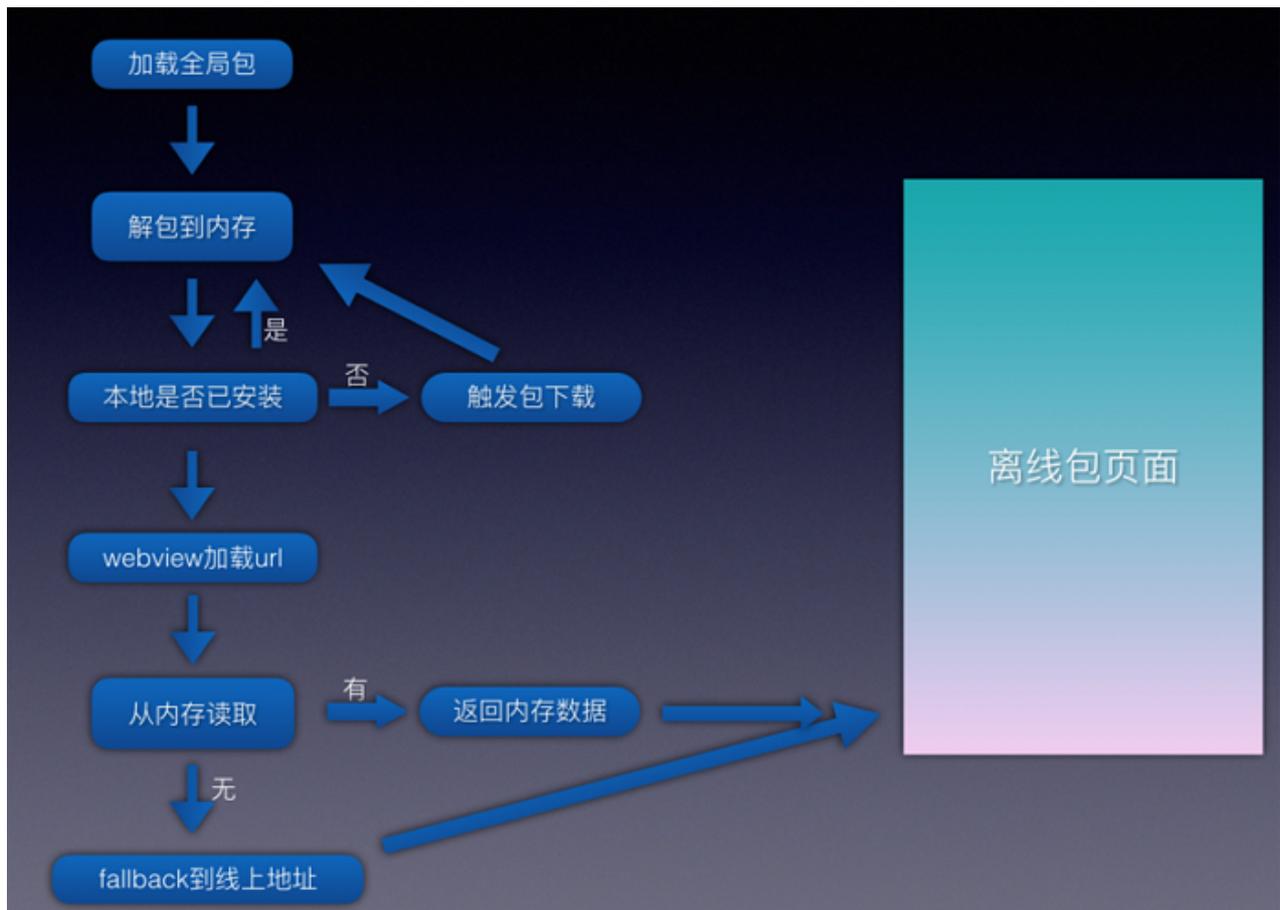
H5 容器会先截获该请求，截获请求后，发生如下情况：

- 如果本地有资源可以满足该请求的话，H5 容器会使用本地资源。
- 如果没有可以满足请求的本地资源，H5 容器会使用线上资源。

因此，无论资源是在本地或者是线上，WebView 都是无感知的。

离线包的下载取决于创建离线包时的配置：

- 如果 **下载时机** 配置为 **仅 WiFi**，则只有在 WiFi 网络时会在后台自动下载离线包。
- 如果 **下载时机** 配置为 **所有网络都下载**，则在非 WiFi 网络时会消耗用户流量自动下载，慎用。
- 如果当前用户点击 APP 时，离线包尚未下载完毕，则会跳转至 fallback 地址，显示在线页面。**fallback** 技术用于应对离线包未下载完毕的场景。每个离线包发布时，都会同步在 CDN 发布一个对应的线上版本，目录结构和离线包结构一致。fallback 地址会随离线包信息下发到本地。在离线包未下载完毕的场景下，客户端会拦截页面请求，转向对应的 CDN 地址，实现在线页面和离线页面随时切换。



离线包运行模式

要打开离线包，您需要完成以下步骤：

1. 请求包信息：从服务端请求离线包信息存储到本地数据库的过程。离线包信息包括离线包的下载地址、离线包版本号等。
2. 下载离线包：把离线包从服务端下载到手机。
3. 安装离线包：下载目录，拷贝到手机安装目录。

虚拟域名

虚拟域名是容器的独特机制，仅对离线应用有效。当页面保存在客户端之后，WebView 如果要访问的话，是通过 file schema 来从本地加载访问的。然而，用户就能在地址栏里直接看到 file 的路径，这就会导致以下问题：

- 用户体验问题：当用户看到了 file 地址，会对暴露的地址产生不安全感 and 不自在。
- 安全性问题：由于 file 协议上直接带上了本地路径，任何用户都可以看到这个文件所在的文件路径，会存在一定的安全隐患。

基于如上问题的考虑，采用虚拟域名的机制而不直接使用 file 路径来访问。虚拟域名是一个符合 URL Scheme 规范的 HTTPS 域名地址，例如 `https://xxxxxxx.h5app.example.com`，虚拟域名的父域名 `example.com` 一定得使用自己注册的域名。

说明：这个域名可以是在互联网上注册过的。但是一般情况下，不建议将虚拟域名配置成互联网一致的域名，这个在判断问题的时候，增加判断的难度，容易出错且不便于日常管理。只要保证其父域名 `example.com` 域名是自己注册的域名即可。现在标准的虚拟域名的格式如下：

```
https://{appid}.h5app.example.com。
```

相关链接

- 生成离线包

2 接入 Android

2.1 添加 Android SDK

H5 容器 SDK 提供了移动端 Web 解决方案，以及良好的外部扩展功能，拥有功能插件化、事件机制、JSAPI 定制和 H5App 推送更新管理能力。要把 H5 容器接入 Android 客户端，需要添加 SDK，并完成相关配置。

前置条件

您已参考 [通用步骤说明](#) 完成基础配置。

操作步骤

在工程中添加 **容器 (NEBULA)** 组件依赖。如果您采用的是 **基于 mPaaS 框架** 的接入方式，则需要分别在 Portal 和 Bundle 工程中添加该组件依赖。更多信息请参考 [管理组件依赖](#) > [增删组件依赖](#)。

2.2 使用 UC 内核

2.2.1 添加 UC SDK

在 Android 应用中接入 UC SDK 能够有效解决各种厂商浏览器的兼容性问题，保持比系统浏览器更低闪退率并且性能卓越。UC SDK 提供安全支持，及时解决安全上的隐患。

申请 UC 内核

申请 UC SDK 的 key :

1. 检查本文起始处的依赖，确保没有错误或者遗漏。
2. 提供应用的应用 ID (application ID) 。
3. 提供应用签名证书指纹的 SHA1 值。执行如下命令获得 SHA1 值：

```
keytool -list -v -keystore keystore路径
```

登录控制台，进入 [提交工单](#) 页面，提供 Android native package 名称和 SHA1 值以获取 UC SDK 的 key。

说明：创建工单时，选择 **问题所属产品** 为 **移动开发平台 mPaaS**（在 **更多产品与服务问题** > **开发者工具** 菜单下），选择问题分类为 **开发框架-开发框架-Android 开发框架接入**。在填写工单信息时，除了必填的个人联系方式外，在 **问题描述** 中填写 **申请 UC SDK 的 key**，用于填写 **Android native package 名称和 SHA1 的值**。

将获取的 Key 填入 Portal 项目的 AndroidManifest.xml 文件中：

```
<meta-data android:name="UCSDKAppKey"android:value="您申请获得的 key"/>
```

说明：UC SDK 的授权信息与 apk 的 **包名** 以及 **签名** 绑定。因此，如果 UCWebView 没有生效，检查签名和包名与申请时使用的信息是否一致。

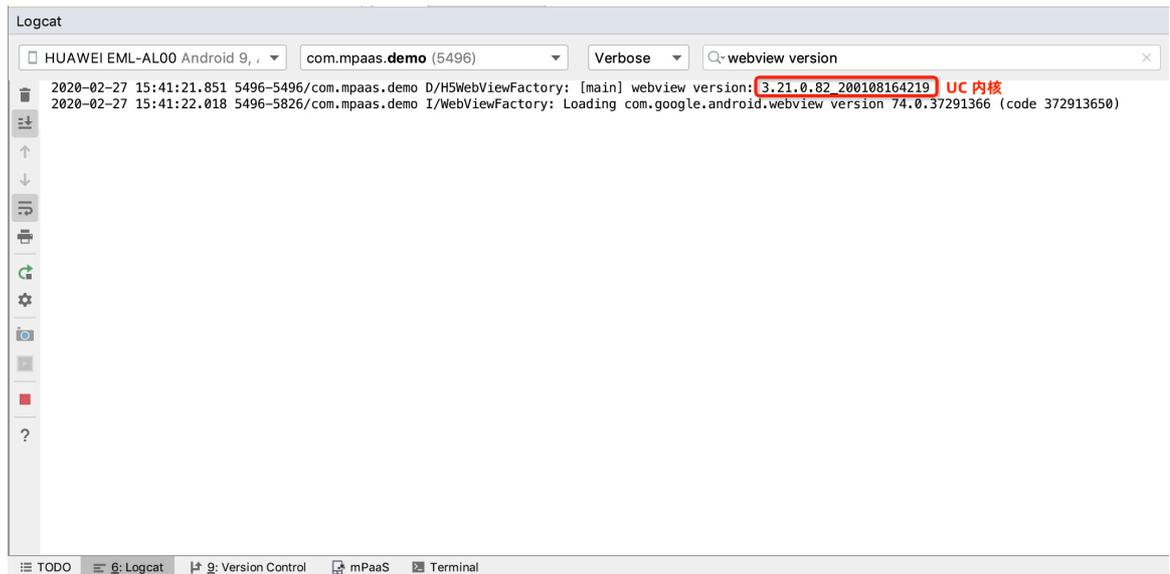
检测 UC 内核是否生效

重新打 debug 包安装后，需要验证 UC 内核是否已经生效。

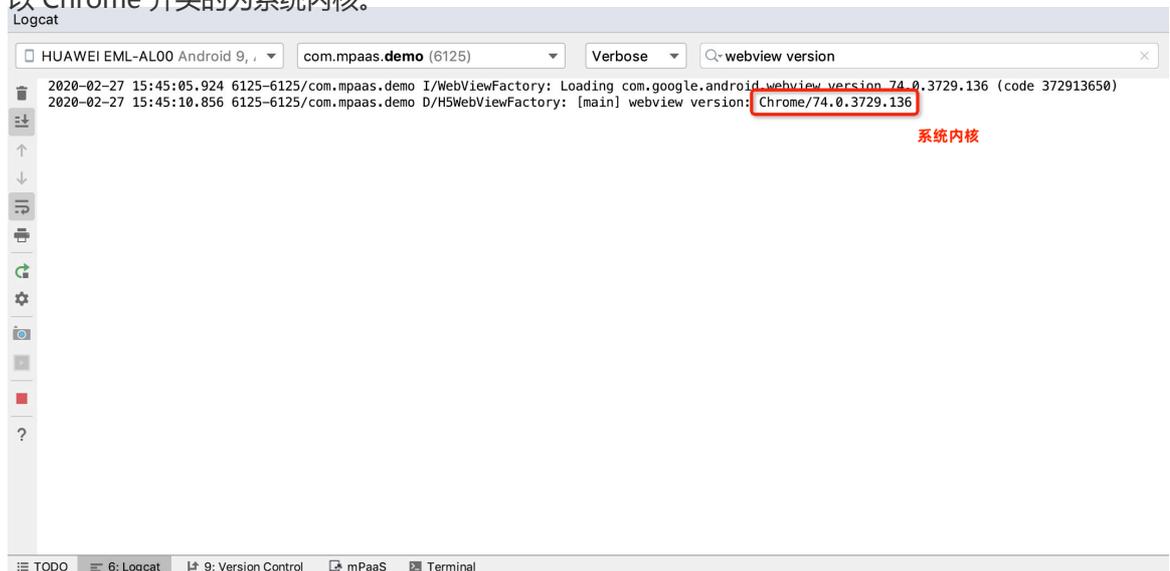
打开任意 H5 页面，在 logcat 日志中过滤 webview version 并搜索。

如下图所示，通常情况下：

- 版本号 (version) 以 2 或 3 开头的为 UC 内核。



- 以 Chrome 开头的为系统内核。



2.2.2 扩展 UC 内核

选择输入键盘

在使用 UC 内核时，您可以自行选择使用系统键盘或 UC 键盘进行输入。

通过为 H5 容器设置 `mp_h5_uc_number_input_use_system` 开关的配置，来控制在 UC 内核下是否使用系统键盘。其中，“YES”表示使用系统键盘，“NO”表示使用 UC 键盘。默认值为“NO”，即使用 UC 内核时，默认使用 UC 键盘进行输入。关于如何设置 H5 容器配置，请您参考 H5 容器配置。

2.3 使用容器

2.3.1 升级说明

10.1.60 变更

MPNebula

```
public static void enableAppVerification(final String publicKey)
```

上述 API 已修改，除调用此 API 之外，仍需要开启验签开关，参见 H5 容器配置。

```
public static void disableAppVerification();
```

上述 API 已废弃，不再支持使用此方法关闭验签，如需关闭验签，参见 H5 容器配置。

H5ExtConfigProvider

已废弃，如需配置 H5 容器开关，参见 H5 容器配置。

2.3.2 使用 SDK

mPaaS Nebula H5 容器提供统一的接口类 MPNebula 来实现 H5 容器及离线包的操作。

调用

调用过程如下：

1. 启动 H5 离线包。

- 启动一个离线包：

```
/**
 * 启动离线包
 *
 * @param appId 离线包 id
 */
public static void startApp(String appId);
```

- 启动一个离线包，带启动参数：

```
/**
 * 启动离线包
 *
 * @param appId 离线包 id
 * @param params 启动参数
 */
public static void startApp(String appId, Bundle params);
```

2. 启动一个在线页面。

- 启动一个在线页面：

```
/**
 * 启动在线 url
 *
 * @param url 在线地址
 */
public static void startUrl(String url)
```

- 启动一个在线页面，带启动参数：

```
/**
 * 启动在线 url
 *
 * @param url 在线地址
 * @param param 启动参数
 */
public static void startUrl(String url, Bundle param);
```

3. 设置自定义 UserAgent。

- 首先您需要实现一个 UA 设置器，如下所示：

```
public class H5UaProviderImpl implements H5UaProvider {
    @Override
    public String getUa(String defaultUaStr) {
        //请不要修改defaultUaStr，或者返回一个不包含defaultUaStr的结果
        return defaultUaStr + " AlipayClient/mPaaS";
    }
}
```

- 然后通过调用设置 UA 接口：

```
/**
 * 设置 UA
 *
 * @param uaProvider UA设置器，需自行实现 getUa 方法
 */
public static void setUa(H5UaProvider uaProvider)
```

- 进行设置：

```
MPNebula.setUa(new H5UaProviderImpl());
```

设置自定义容器 View。

容器提供方法可以设置自定义的标题栏、导航菜单、下拉刷新头部以及 WebView 的承载布局，具体实现，您可参考 代码示例。

首先，您需要先实现一个 View 设置器，如下所示：

```
public class H5ViewProviderImpl implements H5ViewProvider {
    @Override
    public H5WebContentView createWebContentView(Context context) {
        // 此处返回您自定义的 WebView 的承载布局，若返回 null 则使用默认 View
        return null;
    }

    @Override
    public H5TitleView createTitleView(Context context) {
        // 此处返回您自定义的标题栏，若返回 null 则使用默认 View
        return null;
    }

    @Override
    public H5PullHeaderView createPullHeaderView(Context context, ViewGroup viewGroup) {
        // 此处返回您自定义的下拉刷新头部，若返回 null 则使用默认 View
        return null;
    }

    @Override
    public H5NavMenuView createNavMenu() {
        // 此处返回您自定义的导航菜单，若返回 null 则使用默认 View
        return null;
    }
}
```

○ 然后通过 View 设置接口：

```
/**
 * 设置容器相关的自定义 view，如标题栏、菜单栏、web layout、下拉刷新 view 等等
 * @param viewProvider 自定义 view provider
 */
public static void setCustomViewProvider(H5ViewProvider viewProvider);
```

进行设置：

```
MPNebula.setCustomViewProvider(new H5ViewProviderImpl());
```

说明：若您要设置自定义标题栏，请先设置 bundle name，否则可能会导致资源找不到。

```
// 此处必须设置标题栏资源所在的 bundle 名称，如果不设置，会导致资源无法加载导致标题栏无法生效
H5Utils.setProvider(H5ReplaceResourceProvider.class.getName(), new H5ReplaceResourceProvider() {
    @Override
    public String getReplaceResourcesBundleName() {
        return BuildConfig.BUNDLE_NAME;
    }
});
MPNebula.setCustomViewProvider(new H5ViewProviderImpl());
```

将单个容器的视图嵌入到页面。

根据您的实际情况选择以下方法将单个容器的视图 (view) 嵌入到 H5 页面中，接口提供同步方法和异步方法。

说明：使用异步方法不占用主线程，不会影响性能。

- 同步方法：

```
/**
 * 获取 H5 容器的视图 ( view )
 *
 * @param activity 页面上下文
 * @param param 启动参数，内部可包含 appid 或是 url
 * @return H5 容器的视图 ( view )
 */
public static View getH5View(Activity activity, Bundle param);
```

- 异步方法：

```
/**
 * 异步获取 H5 容器的视图 ( view )
 *
 * @param activity 页面上下文
 * @param param 启动参数，内部可包含 appid 或是 url
 * @param h5PageReadyListener 异步回调
 */
public static void getH5ViewAsync(Activity activity, Bundle param, H5PageReadyListener h5PageReadyListener);
```

相关链接

代码示例

2.4 使用离线包

2.4.1 管理离线包

离线包的管理操作包括：预置 H5 应用、利用全局资源包、更新 H5 应用、下载 H5 应用、安装 H5 应用、获取应用信息、以及校验安全签名。

前置条件

- 您已完成接入配置。具体的操作步骤，查看 [添加 SDK](#)。
- 您已经生成离线包。具体的操作步骤，查看 [生成离线包](#)。
- 您已获得读写存储卡的权限。

预置 H5 应用

通常情况下，第一次打开 H5 应用时，离线包可能并未下载完成。此时，只能通过使用 fallback URL 的方式打

开应用。

预置 H5 应用相当于在客户端发布的安装包预先安装可用的 H5 应用。当用户第一次打开预安装应用时，可直接使用离线包资源，提高用户体验。

说明：建议您只预装核心 H5 应用，避免预置使用率不高的应用。

完成以下步骤预置 H5 应用：

1. 从 H5 应用发布后台下载 H5 应用配置的 .json 文件以及需要预置的离线包。
2. 将 .json 文件和离线包添加工程的 asset 目录下。
3. 在应用启动时，调用预置代码安装应用，示例代码如下：

```
MPNebula.loadOfflineNebula("h5_json.json", new  
MPNebulaOfflineInfo("90000000_1.0.0.6.amr","90000000","1.0.0.6"));
```

说明：

- 此方法为阻塞调用，请不要在主线程上调用内置离线包方法。如果内置多个 amr 包，要确保文件已存在，如不存在，会造成其他内置离线包失败。
- 此方法仅能调用一次，若多次调用，仅第一次调用有效。所以需一次性传入所有需预置离线包信息。

利用全局资源包

Nebula 全局资源包解决多个 H5 应用使用同一资源产生的冗余问题。如 React 应用使用 ReactJS 框架代码。您可以将公共资源放入全局资源包，以降低 H5 应用体积。

通常情况下，项目需预置全局资源包，后续更新依然可以通过 H5 应用后台下发。

下方的示例代码指定应用 ID (appId) 为 **66666692** 的离线包作为全局资源包使用，并预置 assets/nebulaPreset/66666692 离线包，其中：

- getCommonResourceAppList：用于告知 H5 容器指定 ID 的离线包将作为全局资源包使用。如果您没有配置此 ID，即使内置该离线包，也不会生效。
- getH5PresetPkg：用于指定内置全局资源包的路径和版本。H5 容器会从指定的 asset 资源目录加载资源包；不过，如果服务端发现更高的版本，该低版本内置包将不会被加载。另外，您也可以使用上文提及的 loadOffLineNebula 方法来预置全局资源包，此种情况下，您无需在此方法中配置内置离线包的路径和版本。

说明：该方法只适用于 H5 全局资源包。

- getTinyCommonApp：仅用于返回小程序的框架资源包 ID，如果您的全局资源包仅被 H5 使用，请不要在此方法中写入该公共资源包 ID。

参考实现类示例代码：

```
public class H5AppCenterPresetProviderImpl implements H5AppCenterPresetProvider {  
private static final String TAG = "H5AppCenterPresetProviderImpl";  
  
// 设置小程序专用资源包
```

```
private static final String TINY_COMMON_APP = "66666692";

// 预置包的asset目录
private final static String NEBULA_APPS_PRE_INSTALL = "nebulaPreset" + File.separator;

// 预置包集合
private static final Map<String, H5PresetInfo> NEBULA_LOCAL_PACKAGE_APP_IDS = new HashMap();

static {

    H5PresetInfo h5PresetInfo2 = new H5PresetInfo();
    // 内置目录的文件名称
    h5PresetInfo2.appId = TINY_COMMON_APP;
    h5PresetInfo2.version = "1.0.0.0";
    h5PresetInfo2.downloadUrl = "";

    NEBULA_LOCAL_PACKAGE_APP_IDS.put(TINY_COMMON_APP, h5PresetInfo2);

}

@Override
public Set<String> getCommonResourceAppList() {
    Set<String> appIdList = new HashSet<String>();
    appIdList.add(getTinyCommonApp());
    return appIdList;
}

@Override
public H5PresetPkg getH5PresetPkg() {
    H5PresetPkg h5PresetPkg = new H5PresetPkg();
    h5PresetPkg.setPreSetInfo(NEBULA_LOCAL_PACKAGE_APP_IDS);
    h5PresetPkg.setPresetPath(NEBULA_APPS_PRE_INSTALL);
    return h5PresetPkg;
}

@Override
public Set<String> getEnableDegradeApp() {
    return null;
}

@Override
public String getTinyCommonApp() {
    return TINY_COMMON_APP;
}

@Override
public InputStream getPresetAppInfo() {
    return null;
}

@Override
public InputStream getPresetAppInfoObject() {
    return null;
}
}
```

然后在应用启动时调用：

```
H5Utils.getH5ProviderManager().setProvider(H5AppCenterPresetProvider.class.getName(), new
H5AppCenterPresetProviderImpl());
```

更新 H5 应用

- 默认情况下，每次打开 H5 应用，Nebula 都会尝试检查是否有可更新的版本。
- 出于服务端压力考虑，该检查有时间间隔限制，默认为 30 分钟。
- 如果想立即检查最新可用版本，调用下方的代码来请求更新。一般情况下，可以在应用启动或者用户登录后调用。

```
MPNebula.updateAllApp(new MpaasNebulaUpdateCallback(){
@Override
public void onResult(final boolean success, final boolean isLimit) {
super.onResult(success, isLimit);
runOnUiThread(new Runnable() {
@Override
public void run() {
AUIToast.makeText(NebulaAppActivity.this,
success ? R.string.update_success : R.string.update_failure, 2000).show();
}
});
}
});
```

下载 H5 应用

MPNebula 接口提供了手动下载 H5 应用的接口：

```
/**
 * 下载离线包
 *
 * @param appId 离线包 id
 * @param mpaasNebulaDownloadCallback 下载回调
 */
public static void downloadApp(final String appId, final MpaasNebulaDownloadCallback
mpaasNebulaDownloadCallback)
```

安装 H5 应用

MPNebula 接口提供了手动安装 H5 应用的接口：

```
/**
 * 安装离线包
 *
 * @param appId 离线包 id
 * @param mpaasNebulaInstallCallback 安装回调
 */
public static void installApp(final String appId, final MpaasNebulaInstallCallback mpaasNebulaInstallCallback)
```

获取应用信息

调用 H5AppProvider 的方法获取 H5 应用的相关信息：

```
H5AppProvider provider = H5Utils.getProvider(H5AppProvider.class.getName());
AppInfo appInfo = provider.getAppInfo("10000000"); //获取应用配置
boolean isInstalled = provider.isInstalled("10000000","1.0.0.0"); //某版本应用是否已经安装
boolean isAvailable = provider.isAvailable("10000000","1.0.0.0"); //某版本应用离线包是否已经下载完成
```

校验安全签名

Nebula 具有离线包签名校验机制，防止恶意程序篡改下载到设备的离线包。

您可通过下列方式开启验签：

通过调用 MPNebula 接口设置验签参数：

说明：

- 请在第一次打开离线包前调用此接口，否则将会导致公钥初始化失败。
- 关于公钥与私钥，参见 [配置离线包 > 上传密钥文件](#)。
- 无论客户端是否开启签名校验，在被判断为 root 的手机上会强制进行签名校验。

```
/**
 * @param publicKey 验签公钥
 */
public static void enableAppVerification(final String publicKey)
```

- 在 10.1.60 及以上版本，需要额外开启容器配置，详情参见 [H5 容器配置](#)。

2.5 自定义 JSAPI

JavaScript API (JSAPI) 是为 H5 应用提供原生能力的接口，您可以利用这些接口使用更多的原生能力和操控能力，提高 H5 应用的用户体验。

H5 容器组件提供以下能力：

- 丰富的内置 JSAPI，实现例如页面 push、pop、标题设置等功能。更多信息查看 [内置 JSAPI](#)。
- 支持用户自定义 JSAPI 和插件功能，扩展业务需求。

本文介绍如何实现自定义 JSAPI 和插件功能。

关于此任务

自定义 JSAPI 插件具备以下特点：

- 为了让各个业务更加灵活地接入 H5 容器，H5 容器提供一种可以给外部业务注册插件配置的机制。
- 业务方使用这种方式注册自定义的外部插件，代码可以放在自己的 bundle 中，全过程不需要 H5 容器介入。使用插件配置的方式注册插件时，H5 容器只在页面调用的时候才初始化对象，并不会立即

生成对象。

- 实现插件和注册插件可以不在同一个 bundle 中，实现依赖解耦。您需要通过 H5Service 动态将插件注入容器。

说明：

- 必须在页面调用 JS 之前注入 JS，一般在静态链接的 Pipeline 中注入。如果过早注入，会出现容器的 bundle 还没加载好（h5service==null）的问题。
- 如果注册的 bundle 不是静态链接，而是懒加载的 bundle 的话，会存在 metaInfo 还没加载，JS 就要调用的问题。

前端页面调用 Native

通过 MPNebula 的注册 JSAPI 接口，注册 JSAPI。

说明： MyJSApiPlugin 的源码可以前往 代码示例 中查看。

- 注册 JSAPI 接口如下：

```
/**
 * 注册自定义 H5 插件 ( JSAPI )
 *
 * @param className 插件类名，需要全路径 ( package + class )
 * @param bundleName bundle 名 ( bundle 名可在主
module/build/intermediates/bundle/META-INF/BUNDLE.MF 中查看，如果该插件写在 portal 工程
中，则 bundleName 需填空字符串" )
 * @param scope 范围，通常为 page
 * @param events 注册的事件
 */
public static void registerH5Plugin(String className, String bundleName, String scope, String[]
events)
```

- 注册示例如下：

```
MPNebula.registerH5Plugin(
MyJSApiPlugin.class.getName(),
BuildConfig.BUNDLE_NAME,
"page",
new String[]{"myapi1", "myapi2", H5Plugin.CommonEvents.H5_PAGE_SHOULD_LOAD_URL}
);
```

前端调用。

通过以下方法使得前端调用自定义 JSAPI：将 event 参数修改成以上插件注册的事件。插件将通过 event.getParam() 获取 JS 的传值，从中解析出数据。

```
AlipayJSBridge.call('myapi2', {
param2: 'World'
});
```

```
function(result) {  
  console.log(result);  
};
```

Native 调用前端页面

H5 容器也可以是 Native 主动调用页面。

以网络变化 JSAPI 为例，页面监听该事件，如下所示：

1. 前端注册监听。

```
document.addEventListener('h5NetworkChange',  
function(e) {  
  alert("网络环境发生变化，可调用 getNetworkType 接口获取详细信息");  
},  
false);
```

2. 客户端监听到网络发生变化时，向页面发送调用事件。

```
JSONObject param = new JSONObject();  
// param 设置自定义的参数  
param.put("data", param);  
H5Page h5Page = h5Service.getTopH5Page();  
if (h5Page != null) {  
  h5Page.getBridge().sendDataWarpToWeb("h5NetworkChange", param, null);  
}
```

使用 Inspect

使用 Chrome 的 inspect 工具，检查调用自定义的 JSAPI 是否有效：

1. 将手机连接电脑，打开电脑的 Chrome 浏览器，输入 `chrome://inspect`，进入调试页面。

使用 mPaaS demo 打开蚂蚁金服金融科技首页。此时，Chrome 的 inspect 页面会变成如下所示：

说明：部分电脑打开 `chrome://inspect` 会白屏，需要升级到 Chrome 最新版本。

点击图中的 **inspect** , 出现如下页面 :

4. 点击页面工具栏上的 **Console** 进入页面调试模式。接下来，您就可以如图例所示调用自己的 API 接口了。

相关链接

- 代码示例
- 常见问题

2.6 自定义功能使用说明

2.6.1 自定义导航栏 (10.1.68(beta))

关于自定义导航栏

Nebula 容器可支持自定义导航栏，您可以自行制定导航栏的样式，例如标题的位置、回按钮的样式等。本文将向您介绍和在 10.1.68(beta) 基线中进行导航栏的自定义。

前提条件

在完整阅读此指南前，需提前知晓以下四点：

1. 小程序和 H5 共用导航栏的实现，在进行自定义导航栏的开发时应将 H5 和小程序使用导航栏的情况都考虑在内，除非确定使用场景不包含 小程序或 H5。
2. 自定义导航栏 **必须符合容器调用的标准流程**，请仔细阅读本文档并按照要求进行开发。
3. 小程序导航栏默认使用内置导航栏，如需开启自定义导航栏，详情请参考 容器配置。
4. 由于导航栏的颜色可以动态设置，为达到最佳体验效果，您应当准备 **两套主题配置** 并根据不同场景进行切换。

接入步骤

1. 继承 **AbstitleView** 抽象类并实现自定义导航栏。

实现 H5ViewProvider，在 createTitleView 方法中创建并返回自定义导航栏实例。

```
public class H5ViewProviderImpl implements H5ViewProvider {
    @Override
    public H5TitleView createTitleView(Context context) {
        return new NewH5TitleViewImpl(context);
    }

    @Override
    public H5NavMenuView createNavMenu() {
        return null;
    }

    @Override
    public H5PullHeaderView createPullHeaderView(Context context, ViewGroup viewGroup) {
        return null;
    }

    @Override
    public H5WebContentView createWebContentView(Context context) {
        return null;
    }
}
```

在合适的地方，比如应用启动时，设置 H5ViewProvider 至容器。

```
MPNebula.setCustomViewProvider(new H5ViewProviderImpl());
```

如果工程是基于 Portal&Bundle 架构，需额外设置。

```
H5Utils.setProvider(H5ReplaceResourceProvider.class.getName(), new H5ReplaceResourceProvider() {
    @Override
    public String getReplaceResourcesBundleName() {
        return BuildConfig.BUNDLE_NAME;
    }
});
```

更多扩展

背景色

```
/**
 * 返回导航栏背景颜色值
 * @return
 */
public abstract int getBackgroundColor();

/**
 * 设置导航栏透明度
 * @param alpha
 */
public abstract void setBackgroundAlphaValue(int alpha);

/**
 * 设置导航栏背景色，不含alpha值
 * @param color
 */
public abstract void setBackgroundColor(int color);

/**
 * 重置导航栏
 */
public abstract void resetTitle();
```

resetTitle 一般在前端页面调用 setTitleColor 的 JSAPI 时被触发。这种情况下，建议您将导航栏的背景以及其他后续提及组件的色值等显示元素重置为默认元素。

标题

副标题当前仅在 H5 场景中支持，如应用不需要副标题，可以不提供副标题实现。

为了使H5页面能够监听 点击标题栏事件，导航栏应该在合适的地方调用 invokeTitleClickEvent方法；监听 点击副标题栏事件，则调用 invokeSubTitleClickEvent 方法。

```
/**
 * 返回主标题文本
 * @return
 */
public abstract String getTitle();

/**
 * 设置主标题文本
 * @param title
 */
public abstract void setTitle(String title);

/**
 * 设置副标题文本
 * @param subTitle
```

```
*/  
public abstract void setSubTitle(String subTitle);  
  
/**  
 * 返回主标题视图  
 * @return  
 */  
public abstract TextView getMainTitleView();  
  
/**  
 * 返回副标题视图  
 * @return  
 */  
public abstract TextView getSubTitleView();
```

左侧控制区

```
/**  
 * 设置关闭按钮可见性  
 * @param visible  
 */  
public abstract void showCloseButton(boolean visible);  
  
/**  
 * 设置返回按钮可见性  
 * @param visible  
 */  
public abstract void showBackButton(boolean visible);  
  
/**  
 * 设置去首页按钮可见性  
 * @param visible  
 */  
public abstract void showBackHome(boolean visible);  
  
/**  
 * 设置标题栏加载提示符可见性  
 * @param visible  
 */  
public abstract void showTitleLoading(boolean visible);
```

关闭按钮



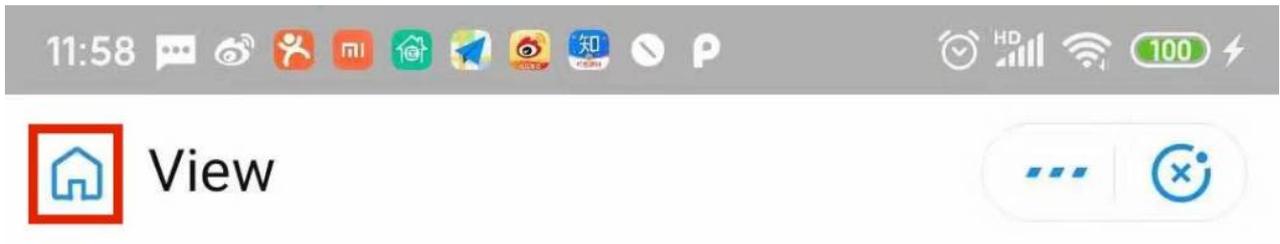
如上图红框区域中所显示，关闭按钮**仅出现在H5场景中**，当在线页面的历史大于1时，容器会调用 `showCloseButton` 方法控件其可见性。此按钮被点击时必须调用 `invokePageCloseEvent` 方法以符合容器行为规范。

返回按钮



上图红框区域中的返回按钮是**必须**要在自定义导航栏上实现的控件，容器调用showBackButton方法控制其可见性。响应返回按钮点击时，开发者必须调用invokePageBackEvent方法以符合容器行为规范。

去首页按钮



去首页按钮**仅**在小程序场景下使用，当跳转小程序的非首页页面时，容器会调用showBackHome方法控制其可见性。响应按钮点击事件时，开发者必须调用invokeHomeClickEvent方法以符合容器行为规范。

加载提示符

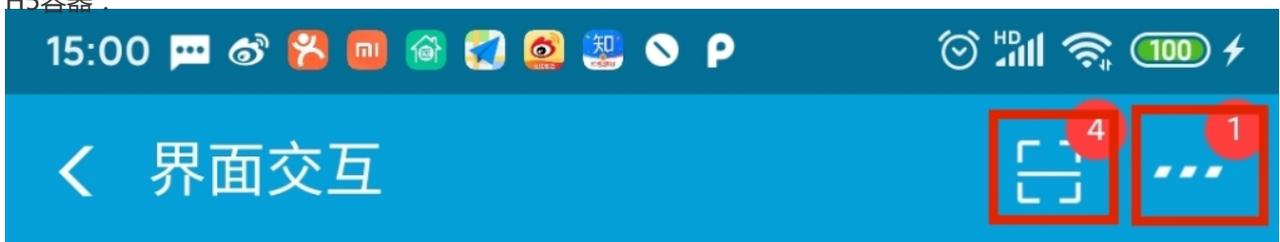


当H5页面或者小程序调用导航栏加载动画API时，容器会调用showTitleLoading方法控制其可见性。

右侧控制区

右侧控制区，也可称为OptionMenu控制区，主要用于提供更多操作给用户。

H5容器：



小程序：



由上图可见，开发者需要为OptionMenu控制区预留两个视图区域，容器操作时会按照索引来操作这两块区域，从右至左，以0开始。

```
public abstract void showOptionsMenu(boolean visible);

public abstract View getOptionsMenuContainer(int index);

public abstract void setOptionsMenu(boolean reset, boolean override, boolean isTinyApp, List<MenuData> menus);
```

容器通过调用 showOptionsMenu 方法来控制 OptionMenu 区域可见性，在某些情况也需要获取该区域的视图进行操作，开发者需要正确实现 getOptionsMenuContainer 方法。

实现 setOptionsMenu 方法，请参考 设置右上角按钮 API 入参进行合理的实现。icontype 为内置按钮样式，仅在 H5 场景中可用，如果接入场景中不会用到则可忽略，否则需要为不同类型配置相应的按钮。redDot 与 icontype 类似，可以选择性接入。

为了使 H5 页面或小程序监听右上角按钮点击事件，开发者需要调用 invokeOptionClickEvent 方法。

小程序场景注意：

1. setOptionsMenu 方法 isTinyApp 参数值必定为 true，可以此区分 H5 和小程序场景。
2. 必须从 **最右侧倒数第二个** 按钮开始设置按钮。

小程序右上角控制区

当处于小程序场景时，最右侧区域需要特殊实现，步骤如下：

1. 继承 AbsTinyOptionsMenuView 抽象类实现自定义控制区。
2. 在合适的地方，比如应用启动时，设置 TinyOptionsMenuViewProvider 至容器。

```
H5Utils.setProvider(TinyOptionsMenuViewProvider.class.getName(), new TinyOptionsMenuViewProvider() {
    @Override
    public AbsTinyOptionsMenuView createView(Context context) {
        return new TinyOptionsMenuView(context);
    }
});
```

按照容器规范要求，开发者必须实现和配置 **更多** 和 **关闭** 按钮视图。

```
public abstract void setOptionsMenuOnClickListener(View.OnClickListener listener);
```

```
public abstract void setCloseButtonOnClickListener(View.OnClickListener listener);

public abstract void setCloseButtonOnLongClickListener(View.OnLongClickListener listener);

public abstract void onStateChanged(TinyAppActionState currentState);

public abstract View getView();
```

容器会调用上面三个方法来设置合理的响应回调，开发者必须让指定的视图设置该响应回调。
onStateChanged 方法在定位、蓝牙场景中会被调用，举例来说，当小程序正在使用定位服务的时，容器会调用此方法，开发者可作出响应，下面是样式效果。



简单示例代码

```
public class TinyOptionsMenuView extends AbsTinyOptionsMenuView {

    private View container;

    private ImageView ivMore;

    private View ivClose;

    private Context context;

    public TinyOptionsMenuView(Context context) {
        this.context = context;
        ViewGroup parent = null;
        if (context instanceof Activity) {
            parent = (ViewGroup) ((Activity) context).findViewById(android.R.id.content);
        }
        container = LayoutInflater.from(context).inflate(R.layout.layout_tiny_right, parent, false);
        ivClose = container.findViewById(R.id.close);
        ivMore = (ImageView) container.findViewById(R.id.more);
    }

    @Override
    public View getView() {
        return container;
    }

    @Override
    public void setOptionsMenuOnClickListener(View.OnClickListener onClickListener) {
        ivMore.setOnClickListener(onClickListener);
    }
}
```

```

@Override
public void setCloseButtonOnClickListener(View.OnClickListener onClickListener) {
    ivClose.setOnClickListener(onClickListener);
}

@Override
public void setCloseButtonOnLongClickListener(View.OnLongClickListener onLongClickListener) {
    ivClose.setOnLongClickListener(onLongClickListener);
}

@Override
public void onStateChanged(TinyAppActionState state) {
    if (state == null) {
        ivMore.setImageDrawable(context.getResources().getDrawable(R.drawable.icon_more));
    } else if (state.getAction().equals(TinyAppActionState.ACTION_LOCATION)) {
        ivMore.setImageDrawable(context.getResources().getDrawable(R.drawable.icon_miniprogram_location));
    }
}
}
}

```

主题变更

不同的小程序或 H5 应用可能会设置不同的导航栏背景色，因此为了体验考虑，在导航栏背景色发生变化的时候，也需要对导航栏上的其他元素进行调整，比如右上角控制区。

在导航栏的扩展上，右上角控制区和导航栏分别为不同的组件，所以我们提供接口来方便开发者能够让右上角控制区能够及时响应导航栏的变化。

AbsTinyOptionMenuView 提供 onTitleChanged 方法供 override 来响应导航栏变化。当此方法被调用时，可通过传入 H5TitleView 对象来获取导航栏的信息，如背景色。也可以使用 AbsTinyOptionMenuView 提供的 getTitleBar 方法直接获取导航栏对象。另外，您可以将 H5TitleView 转换成自己实现的导航栏对象，因为 AbsTitleView 实现了 H5TitleView 接口类，这样能够获得更多的导航栏信息。

```
protected void onTitleChange(H5TitleView title)
```

您需要主动调用 AbsTitleView 提供的 notifyTitleBarChanged 方法触发 onTitleChange 被调用，比如在对导航栏对象设置背景色时。举例说明：

```

package com.mpaas.demo.nebula;

public class NewH5TitleViewImpl extends AbsTitleView {

    @Override
    public void setBackgroundAlphaValue(int i) {
        content.getContentBgView().setAlpha(i);
        notifyTitleBarChanged();
    }

    @Override
    public void setBackgroundColor(int i) {
        content.getContentBgView().setColor(i);
    }
}

```

```

notifyTitleBarChanged();
}

@Override
public void resetTitle() {
content.getContentBgView().setColor(context.getResources().getColor(R.color.h5_default_titlebar_color));
notifyTitleBarChanged();
}
}

```

上述示例代码调用 `notifyTitleBarChange` 方法时，`AbsTinyOptionsMenuView` 的子类对象可能并没有初始化完毕，因此建议覆写 `setH5Page` 方法，获取导航栏信息并判定当前主题，示例如下：

```

public class TinyOptionsMenuView extends AbsTinyOptionsMenuView {

@Override
public void setH5Page(H5Page h5Page) {
super.setH5Page(h5Page);
// title becomes available from here.
if (getTitleBar().getBackgroundColor() == -1) {
bgView.setBackgroundColor(Color.RED);
}
}
}
}

```

2.6.2 自定义标题栏 (10.1.60)

如果有自定义 H5 容器标题栏的需求，参考下方示例代码实现：

10.1.60

H5TitleViewImpl.java

```

package com.mpaas.demo.nebula;

import android.app.Activity;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageButton;
import android.widget.LinearLayout;
import android.widget.TextView;

import com.alibaba.fastjson.JSONArray;
import com.alibaba.fastjson.JSONObject;
import com.alipay.mobile.h5container.api.H5Page;
import com.alipay.mobile.h5container.api.H5Param;

```

```
import com.alipay.mobile.h5container.api.H5Plugin;
import com.alipay.mobile.nebula.util.H5Log;
import com.alipay.mobile.nebula.util.H5StatusBarUtils;
import com.alipay.mobile.nebula.util.H5Utils;
import com.alipay.mobile.nebula.view.H5TitleBarFrameLayout;
import com.alipay.mobile.nebula.view.H5TitleView;
import com.alipay.mobile.nebula.view.IH5TinyPopupMenu;

import java.util.ArrayList;
import java.util.List;

/**
 * Created by omg on 2018/7/23.
 */

public class H5TitleViewImpl implements H5TitleView, View.OnClickListener {

    private static final String TAG = "H5TitleViewImpl";

    private Context mContext;

    private H5TitleBarFrameLayout contentView;

    private String title;

    // 此处定义标题栏的基本控件
    private TextView mTitleView;

    private TextView mSubTitleView;

    private View mCloseButton;

    private View mBackButton;

    private View vDivider;

    private View hDivider;

    private View statusBarAdjustView;

    /**
     * ===== OptionMenu的各种View Start =====
     */
    // OptionMenu的Container
    public View h5NavOptions;
    public View h5NavOptions1;
    public List<View> h5NavOptionsList = new ArrayList<View>();

    // ---- OptionMenu 三种形态 Start ---- //
    // 1. OptionType.MENU(默认) - 默认的 Option 按钮
    public TextView btMenu;
    public TextView btMenu1;
    public List<TextView> btMenuList = new ArrayList<TextView>();

    // 2. OptionType.ICON - 通过 setOptionMenu 手动设置的 icon
    public ImageButton btIcon;
```

```
public ImageButton btIcon1;
public List<ImageButton> btIconList = new ArrayList<ImageButton>();

// 3. OptionType.TEXT - 文字
public TextView btText;
public TextView btText1;
public List<TextView> btTextList = new ArrayList<TextView>();
// ---- OptionMenu 三种形态 Over ---- //

// Web 页面的实例接口类，可用于控制 web 行为
private H5Page h5Page;

private int visibleOptionNum = 0;
private IH5TinyPopupMenu h5TinyPopupMenu;

/**
 * H5 容器标题栏构造方法
 * 注意：标题栏布局 xml 文件必须以 H5TitleBarFrameLayout 作为根节点
 * @param context
 */
public H5TitleViewImpl(Context context) {
    mContext = context;
    ViewGroup parent = null;
    if (context instanceof Activity) {
        parent = (ViewGroup) ((Activity) mContext).findViewById(android.R.id.content);
    }
    contentView = (H5TitleBarFrameLayout) LayoutInflater.from(context).inflate(R.layout.h5_navigation_bar, parent,
    false);
    contentView.getContentBgView().setColor(context.getResources().getColor(R.color.h5_default_titlebar_color));
    mTitleView = (TextView) contentView.findViewById(R.id.h5_tv_title);
    mTitleView.setOnClickListener(this);
    mSubTitleView = (TextView) contentView.findViewById(R.id.h5_tv_subtitle);
    mSubTitleView.setOnClickListener(this);
    mCloseButton = contentView.findViewById(R.id.h5_nav_close);
    mCloseButton.setOnClickListener(this);
    mBackButton = contentView.findViewById(R.id.h5_tv_nav_back);
    mBackButton.setOnClickListener(this);
    vDivider = contentView.findViewById(R.id.h5_v_divider);
    hDivider = contentView.findViewById(R.id.h5_h_divider_intitle);
    h5NavOptions = contentView.findViewById(R.id.h5_nav_options);
    h5NavOptions1 = contentView.findViewById(R.id.h5_nav_options1);
    statusBarAdjustView = contentView.findViewById(R.id.h5_status_bar_adjust_view);

    btIcon = (ImageButton) contentView.findViewById(R.id.h5_bt_image);
    btText = (TextView) contentView.findViewById(R.id.h5_bt_text);
    btMenu = (TextView) contentView.findViewById(R.id.h5_bt_options);

    btIcon1 = (ImageButton) contentView.findViewById(R.id.h5_bt_image1);
    btText1 = (TextView) contentView.findViewById(R.id.h5_bt_text1);
    btMenu1 = (TextView) contentView.findViewById(R.id.h5_bt_options1);

    //add view to list
    h5NavOptionsList.add(h5NavOptions);
    h5NavOptionsList.add(h5NavOptions1);

    btIconList.add(btIcon);
```

```
btIconList.add(btIcon1);

btTextList.add(btText);
btTextList.add(btText1);

btMenuList.add(btMenu);
btMenuList.add(btMenu1);

btText.setOnClickListener(this);
btIcon.setOnClickListener(this);
btText1.setOnClickListener(this);
btIcon1.setOnClickListener(this);
btMenu.setOnClickListener(this);
btMenu1.setOnClickListener(this);

}

/**
 * 容器调用此方法获得主标题内容
 */
@Override
public String getTitle() {
return title;
}

/**
 * 容器调用此方法设置主标题内容
 */
@Override
public void setTitle(String s) {
title = s;
mTitleView.setText(s);
}

/**
 * 容器调用此方法设置副标题内容
 */
@Override
public void setSubTitle(String s) {
mSubTitleView.setVisibility(View.VISIBLE);
mSubTitleView.setText(s);
}

/**
 * 暂时忽略，可不实现
 */
@Override
public void setImgTitle(Bitmap bitmap) {

}

/**
 * 暂时忽略，可不实现
 */
@Override
public void setImgTitle(Bitmap bitmap, String s) {
```

```
}

/**
 * 容器设置是否显示关闭按钮
 */
@Override
public void showCloseButton(boolean b) {
    mCloseButton.setVisibility(b ? View.VISIBLE : View.GONE);
}

/**
 * 容器获取标题栏 View
 */
@Override
public View getContentView() {
    return contentView;
}

/**
 * 容器获取标题栏背景用于设置背景色
 */
@Override
public ColorDrawable getContentBgView() {
    return contentView.getContentBgView();
}

/**
 * 容器获取主标题 View
 * 不可为空
 */
@Override
public TextView getMainTitleView() {
    return mTitleView;
}

/**
 * 容器获取副标题 View
 * 不可为空
 */
@Override
public TextView getSubTitleView() {
    return mSubTitleView;
}

/**
 * 设置是否展示返回按钮
 */
@Override
public void showBackButton(boolean b) {
    mBackButton.setVisibility(b ? View.VISIBLE : View.GONE);
}

/**
 * 设置是否显示右上角菜单项
 */
```

```
@Override
public void showOptionsMenu(boolean isShow) {
    if (isShow) {
        switch (visibleOptionNum) {
            case 1:
                h5NavOptions.setVisibility(View.VISIBLE);
                break;
            case 2:
                h5NavOptions.setVisibility(View.VISIBLE);
                h5NavOptions1.setVisibility(View.VISIBLE);
                break;
        }
    } else {
        h5NavOptions.setVisibility(View.GONE);
        h5NavOptions1.setVisibility(View.GONE);
    }
}

/**
 * 设置右上角菜单项展示类型，可为图标、文字
 */
@Override
public void setOptionType(H5Param.OptionType optionType) {
    setOptionType(optionType, 0, true);
}

/**
 * 设置右上角菜单项展示类型，可为图标、文字
 * @param byIndex 是否只对某个菜单项设置展示类型
 */
@Override
public void setOptionType(H5Param.OptionType type, int num, boolean byIndex) {
    boolean icon = false;
    boolean text = false;
    boolean menu = false;
    if (type == H5Param.OptionType.ICON) {
        icon = true;
    } else if (type == H5Param.OptionType.TEXT) {
        text = true;
    } else if (type == H5Param.OptionType.MENU) {
        menu = true;
    }
    ctrlbtText(num, text ? View.VISIBLE : View.GONE, byIndex);
    ctrlbtIcon(num, icon ? View.VISIBLE : View.INVISIBLE, byIndex);
    ctrlbtMenu(num, menu ? View.VISIBLE : View.INVISIBLE, byIndex);
}

//view visible control
private boolean isOutOfBounds(int num, int length) {
    if (length == 0) {
        return true;
    }
    return length < num;
}

private void ctrlbtText(int num, int visible, boolean byIndex) {
```

```
if (isOutOfBounds(num, btTextList.size())) {
    return;
}
if (byIndex) {
    btTextList.get(num).setVisibility(visible);
} else {
    for (int i = 0; i < num; i++) {
        btTextList.get(i).setVisibility(visible);
    }
}

private void ctrlbtIcon(int num, int visible, boolean byIndex) {
    if (isOutOfBounds(num, btIconList.size())) {
        return;
    }
    if (byIndex) {
        btIconList.get(num).setVisibility(visible);
    } else {
        for (int i = 0; i < num; i++) {
            btIconList.get(i).setVisibility(visible);
        }
    }
}

private void ctrlbtMenu(int num, int visible, boolean byIndex) {
    if (isOutOfBounds(num, btMenuList.size())) {
        return;
    }
    if (byIndex) {
        btMenuList.get(num).setVisibility(visible);
    } else {
        for (int i = 0; i < num; i++) {
            btMenuList.get(i).setVisibility(visible);
        }
    }
}

/**
 * 设置是否在标题栏上显示加载状态，可自行选择实现方式
 */
@Override
public void showTitleLoading(boolean b) {

}

/**
 * 可忽略
 */
@Override
public void showTitleDisclaimer(boolean b) {

}

// 设置右上角按钮图标
@Override
```

```
public void setBtIcon(Bitmap btIcon, int index) {
    if (isOutOfBounds(index, btIconList.size())) {
        return;
    }
    btIconList.get(index).setImageBitmap(btIcon);
}

@Override
public void setH5Page(H5Page h5Page) {
    this.h5Page = h5Page;
}

/**
 * 根据 JS 传递过来的参数设置右上角菜单
 */
@Override
public void setOptionsMenu(JSONObject params) {
    boolean reset = H5Utils.getBoolean(params, "reset", false);
    boolean override = H5Utils.getBoolean(params, "override", false);
    JSONArray menus = H5Utils.getJSONArray(params, "menus", null);
    if (reset) {
        h5NavOptions1.setVisibility(View.GONE);
        setOptionType(H5Param.OptionType.MENU, 0, true);
        visibleOptionNum = 1;
        return;
    }
    if (menus != null && !menus.isEmpty()) {
        visibleOptionNum = 0;
        if (override) {
            int menuSize = menus.size() > 2 ? 2 : menus.size();
            for (int i = 0; i < menuSize; i++) {
                // h5NavOptionsList.get(i).setVisibility(View.VISIBLE);
                JSONObject menuItem = menus.getJSONObject(i);
                setOptionsMenuInternal(menuItem, i);
                visibleOptionNum++;
            }
        } else {
            visibleOptionNum = 2;
            // h5NavOptionsList.get(1).setVisibility(View.VISIBLE);
            JSONObject menuItem = menus.getJSONObject(0);
            setOptionsMenuInternal(menuItem, 1);
        }
    } else {
        setOptionsMenuInternal(params, 0);
        visibleOptionNum = 1;
    }
}

private void setOptionsMenuInternal(JSONObject params, int index) {
    String title = H5Utils.getString(params, "title");
    String icon = H5Utils.getString(params, "icon");
    String icontype = H5Utils.getString(params, "icontype");
    String contentDesc = H5Utils.getString(params, "contentDesc");
    String colorText = H5Utils.getString(params, "color");

    // default white color
```

```
int color = 0xFF108ee9;
if (!TextUtils.isEmpty(colorText)) {
    try {
        color = Color.parseColor(colorText);
    } catch (Throwable ignore) {
        //can not find logutil
    }
    color = 0xFF000000 | color;
    btTextList.get(index).setTextColor(color);
} else {
    int currentColor = mView.getCurrentTextColor();
    currentColor = 0xFF000000 | currentColor;
    H5Log.d(TAG, "setOptionsMenuInternal currentColor is" + currentColor);
    if (currentColor != 0xFF111111) {
        btText.setTextColor(0xFFFFFFFF);
        btText1.setTextColor(0xFFFFFFFF);
    } else {
        btText.setTextColor(0xFF108ee9);
        btText1.setTextColor(0xFF108ee9);
    }
}

if (!TextUtils.isEmpty(title)) {
    title = title.trim();
    btTextList.get(index).setText(title);
    setOptionType(H5Param.OptionType.TEXT, index, true);
    btTextList.get(index).setContentDescription(title);
} else if (!TextUtils.isEmpty(icon) || !TextUtils.isEmpty(iconType)) {
    if (!TextUtils.isEmpty(contentDesc)) {
        btIconList.get(index).setContentDescription(contentDesc);
    }
}

/**
 * 容器获取返回按钮和标题内容之间的分割线
 * 可返回空
 */
@Override
public View getDivider() {
    return vDivider;
}

/**
 * 容器获取标题栏和 Web 页面间的分割线
 * 不可为空
 */
@Override
public View getHdividerInTitle() {
    return hDivider;
}

/**
 * 容器获取下拉菜单弹出位置的锚点View
 */
@Override
```

```
public View getPopAnchor() {
    return btMenu;
}

/**
 * 容器重置标题栏背景色
 */
@Override
public void resetTitleColor(int color) {

}

/**
 * 暂时忽略
 */
@Override
public void switchToWhiteTheme() {

}

/**
 * 暂时忽略
 */
@Override
public void switchToBlueTheme() {

}

/**
 * 容器页面销毁时触发，此处可释放被引用的 View
 */
@Override
public void releaseViewList() {
    if (h5NavOptionsList != null) {
        h5NavOptionsList.clear();
    }
    if (btIconList != null) {
        btIconList.clear();
    }
    if (btTextList != null) {
        btTextList.clear();
    }
    if (btMenuList != null) {
        btMenuList.clear();
    }
}

/**
 * 容器设置沉浸式标题栏颜色
 */
@Override
public void openTranslucentStatusBarSupport(int color) {
    if (H5StatusBarUtils.isSupport()) {
        int statusBarHeight = H5StatusBarUtils.getStatusBarHeight(mContext);
```

```
if (statusBarHeight == 0) { //保护，万一有 rom 没办法拿到状态栏高度的话，则在这里不生效。
return;
}
LinearLayout.LayoutParams layoutParams =
(LinearLayout.LayoutParams) statusBarAdjustView.getLayoutParams();
layoutParams.height = statusBarHeight;
statusBarAdjustView.setLayoutParams(layoutParams);
statusBarAdjustView.setVisibility(View.VISIBLE);

try {
H5StatusBarUtils.setTransparentColor((Activity) mContext, color);
} catch (Exception e) {
H5Log.e(TAG, e);
}
}
}

/**
* 暂时忽略
*/
@Override
public void switchToTitleBar() {

}

/**
* 暂时忽略
*/
@Override
public View setTitleBarSearch(Bundle bundle) {
return null;
}

/**
* 暂时忽略
*/
@Override
public void setBackCloseBtnImage(String s) {

}

/**
* 设置标题栏字体颜色
*/
@Override
public void setTitleTxtColor(int i) {
mTitleView.setTextColor(i);
mSubTitleView.setTextColor(i);
}

/**
* 容器获取右上角菜单 View, 必须是ViewGroup及子类
*/
@Override
public View getOptionsMenuContainer() {
return h5NavOptions;
}
```

```
}

/**
 * 容器根据位置获取右上角菜单 View，必须是ViewGroup及子类
 */
@Override
public View getOptionsMenuContainer(int index) {
    switch (index) {
        case 0:
            return h5NavOptions;
        case 1:
            return h5NavOptions1;
        default:
            return getOptionsMenuContainer();
    }
}

/**
 * 暂时忽略
 */
@Override
public void setIH5TinyPopupMenu(IH5TinyPopupMenu tinyPopupMenu) {
    this.h5TinyPopupMenu = tinyPopupMenu;
}

/**
 * 暂时忽略
 */
@Override
public IH5TinyPopupMenu getH5TinyPopupMenu() {
    return null;
}

/**
 * 暂时忽略
 */
@Override
public void setTitleView(View view) {

}

/**
 * 暂时忽略
 */
@Override
public void initTitleSegControl(JSONObject jsonObject) {

}

/**
 * 暂时忽略
 */
@Override
public void enableTitleSegControl(boolean b) {

}
}
```

```
/**
 * 暂时忽略
 */
@Override
public void enableBackButtonBackground(boolean b) {

}

/**
 * 处理标题栏上不同控件的点击事件
 * 如果 JS 需要收到事件，需按照该代码中呈现的方式发送事件给 JS
 * 比如点击返回按钮，则发送 H5Plugin.CommonEvents.H5_TOOLBAR_BACK 事件
 */
@Override
public void onClick(View view) {
if (h5Page == null) {
return ;
}

String eventName = null;
JSONObject data = null;

if (view == mBackButton) {
eventName = H5Plugin.CommonEvents.H5_TOOLBAR_BACK; // 发送后退事件
} else if (view == mCloseButton) {
eventName = H5Plugin.CommonEvents.H5_TOOLBAR_CLOSE; // 发送关闭页面事件
} else if (view.equals(btIcon) || view.equals(btText)) {
eventName = H5Plugin.CommonEvents.H5_TITLEBAR_OPTIONS;
data = new JSONObject();
data.put("index", 0);
} else if (view.equals(btIcon1) || view.equals(btText1)) {
eventName = H5Plugin.CommonEvents.H5_TITLEBAR_OPTIONS;
data = new JSONObject();
data.put("index", 1);
} else if (view.equals(btMenu) || view.equals(btMenu1)) {
eventName = H5Plugin.CommonEvents.H5_TITLEBAR_OPTIONS;
data = new JSONObject();
data.put("fromMenu", true);
data.put("index", view.equals(btMenu) ? 0 : 1);
} else if (view.equals(mTitleView)) {
eventName = H5Plugin.CommonEvents.H5_TITLEBAR_TITLE;
} else if (view.equals(mSubTitleView)) {
eventName = H5Plugin.CommonEvents.H5_TITLEBAR_SUBTITLE;
}

if (!TextUtils.isEmpty(eventName)) {
h5Page.sendEvent(eventName, data);
}
}
}
```

h5_navigation_bar.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<com.alipay.mobile.nebula.view.H5TitleBarFrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/h5_title_bar"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:background="@android:color/transparent" >
```

```
<LinearLayout
android:id="@+id/h5_rl_title_bar"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:background="@android:color/transparent"
android:orientation="vertical" >
```

```
<View
android:id="@+id/h5_status_bar_adjust_view"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:visibility="gone"/>
```

```
<RelativeLayout
android:id="@+id/h5_title_bar_layout"
android:layout_width="match_parent"
android:layout_height="48dp" >
```

```
<ImageButton
android:id="@+id/h5_tv_nav_back"
android:layout_width="48dp"
android:layout_height="match_parent"
android:layout_alignParentLeft="true"
android:layout_centerVertical="true"
android:background="@android:color/transparent"
android:scaleType="centerInside"
android:padding="12dp"
android:src="@drawable/back"/>
```

```
<ImageButton
android:id="@+id/h5_nav_close"
android:layout_width="48dp"
android:layout_height="match_parent"
android:layout_centerVertical="true"
android:padding="12dp"
android:layout_marginLeft="-6dp"
android:layout_toRightOf="@+id/h5_tv_nav_back"
android:background="@android:color/transparent"
android:clickable="true"
android:scaleType="centerInside"
android:src="@drawable/close"/>
```

```
<View
android:id="@+id/h5_v_divider"
android:layout_width="0.7dp"
android:layout_height="24dp"
android:layout_centerVertical="true"
android:layout_gravity="center"
```

```
android:layout_marginRight="12dp"  
android:layout_toRightOf="@+id/h5_nav_close"  
tools:ignore="ContentDescription"/>
```

```
<RelativeLayout  
android:id="@+id/h5_rl_title"  
android:layout_width="match_parent"  
android:layout_height="wrap_content"  
android:layout_centerVertical="true"  
android:gravity="center">
```

```
<LinearLayout  
android:id="@+id/h5_ll_title"  
android:layout_width="wrap_content"  
android:layout_height="match_parent"  
android:gravity="center"  
android:orientation="vertical">
```

```
<FrameLayout  
android:layout_width="wrap_content"  
android:layout_height="wrap_content">
```

```
<TextView  
android:id="@+id/h5_tv_title"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:ellipsize="end"  
android:singleLine="true"  
android:textColor="@android:color/white"  
android:textSize="16dp"/>
```

```
<ImageView  
android:id="@+id/h5_tv_title_img"  
android:layout_width="wrap_content"  
android:layout_height="36dp"  
android:scaleType="centerInside"  
android:visibility="gone"/>  
</FrameLayout>
```

```
<TextView  
android:id="@+id/h5_tv_subtitle"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:singleLine="true"  
android:textColor="@android:color/white"  
android:textSize="12dp"  
android:visibility="gone"  
tools:visibility="visible"/>  
</LinearLayout>
```

```
</RelativeLayout>
```

```
<!-- optionmenu0-->
```

```
<FrameLayout  
android:id="@+id/h5_nav_options"
```

```
android:layout_width="wrap_content"
android:layout_height="fill_parent"
android:layout_alignParentRight="true"
android:layout_centerVertical="true"
android:layout_marginLeft="6dp">

<ImageButton
android:id="@+id/h5_bt_image"
android:layout_width="32dp"
android:layout_height="32dp"
android:layout_gravity="center|right"
android:layout_marginRight="12dp"
android:background="@android:color/transparent"
android:padding="4dp"
android:scaleType="fitCenter"
android:visibility="gone"/>

<TextView
android:id="@+id/h5_bt_text"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center|right"
android:layout_marginRight="12dp"
android:background="@null"
android:ellipsize="end"
android:maxLength="8"
android:singleLine="true"
android:textColor="#ffffff"
android:textSize="16dp"/>

<TextView
android:id="@+id/h5_bt_options"
android:layout_width="48dp"
android:layout_height="match_parent"
android:background="@null"
android:gravity="center"
android:textSize="23dp"
android:visibility="gone"
tools:visibility="gone"/>
</FrameLayout>

<!-- optionmenu1-->
<FrameLayout
android:id="@+id/h5_nav_options1"
android:layout_width="wrap_content"
android:layout_height="fill_parent"
android:layout_centerVertical="true"
android:layout_marginLeft="6dp"
android:layout_toLeftOf="@id/h5_nav_options"
android:visibility="gone"
tools:visibility="visible">

<ImageButton
android:id="@+id/h5_bt_image1"
android:layout_width="32dp"
android:layout_height="32dp"
```

```
android:layout_gravity="center|right"
android:layout_marginRight="12dp"
android:background="@android:color/transparent"
android:padding="4dp"
android:scaleType="fitCenter"
android:visibility="gone"/>

<TextView
android:id="@+id/h5_bt_text1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center|right"
android:layout_marginRight="12dp"
android:background="@null"
android:ellipsize="end"
android:maxLength="8"
android:singleLine="true"
android:textColor="#108ee9"
android:textSize="16dp"/>

</FrameLayout>
</RelativeLayout>

<View
android:id="@+id/h5_h_divider_intitle"
android:layout_width="match_parent"
android:layout_height="1px"
android:background="@android:color/white"
android:visibility="gone"/>

</LinearLayout>

</com.alipay.mobile.nebula.view.H5TitleBarFrameLayout>
```

2.6.3 H5 容器配置

H5 容器内有许多开关配置，通过修改开关配置，能够改变容器的特定行为，比如可以通过验签配置来开启或关闭离线包签名校验。

修改开关配置有以下三种方式：

- 内置 custom_config.json 至 portal 工程或应用主工程 assets 目录下的 config 文件夹，该方法仅适用于 10.1.60 及以上版本。custom_config.json 的文件格式如下：

```
[
{
  "value":"NO",
  "key":"h5_shouldverifyapp"
},
{
  "value":"0",
  "key":"TSBS"
}
]
```

```
]
```

- 使用 H5ExtConfigProvider 在代码中配置开关，该方法仅适用于 10.1.60 以下版本。
H5ExtConfigProvider 使用说明如下：

```
public class H5ExtConfigProviderImpl implements H5ExtConfigProvider {
    @Override
    public String getConfig(String key) {
        if ("h5_shouldverifyapp".equalsIgnoreCase(key)) {
            return "YES";
        } else if ("TSBS".equalsIgnoreCase(key)) {
            return "0";
        }
        return null;
    }
}

// 建议在启动时调用，全局只生效一个 H5ExtConfigProvider 的实例，以最后设置的实例为准
H5Utils.setProvider(H5ExtConfigProvider.class.getName(), new H5ExtConfigProviderImpl());
```

- 通过 MDS 平台下发开关配置，参见 开关配置管理。

容器开关列表

您可通过下表中的开关，来自定义是否使用对应功能。

开关名称	用途	说明	默认值
h5_shouldverifyapp	开启或关闭验签。建议线上开启，当手机被认为是 root 手机时，会强制开启验签，此时开关配置不生效。	YES 表示开启，NO 表示关闭。	YES
TSBS	是否使用沉浸式标题栏，仅适用于 Android。	"1" 表示使用，"0" 表示不使用。 注意 ：此处的 "1" 和 "0" 为字符串形式。	0
h5_remote_debug_host	真机调试远程服务器地址。	<ul style="list-style-type: none"> 如果配置，表示开启了远程真机调试。调试用服务器地址需要配置给 h5_remote_debug_host。 如果不配置，即在代码中不体现，表示不使用远程真机进行调试，也无默认值。 	-
androidFallbackNetwork	是否采用 mPaaS 网络库方式加载 fallback 资源。	YES 表示使用 mpaas 网络库加载 fallback 资源，NO 表示使用系统网络库加载 fallback 资源。	YES
mp_h5_push_window_use_activity	调用 pushWindow 时是否强制启动新的 Activity。	YES 表示启用，其他值表示不启用。	NO
mp_ta_showOptionsMenu	是否显示小程序右上角选项菜单。 说明 ：此配置仅在发布小程序时选择是否显示右上角菜单时有效。	YES 表示显示，其他值表示不显示。	NO
mp_ta_showShareMenuItem	是否显示小程序右上角选项菜单中的分享选项。	YES 表示显示，其他值表示不显示。	NO

mp_ta_use_organal_mini_navigationbar	小程序是否使用内置导航栏。	YES 表示使用，其他值表示不使用。	YES
h5_CORSWhiteList	域名白名单，该域名下的离线资源可被跨域访问。 说明： 对于在线请求的资源，仍需资源服务端开启正确的跨域设置。	内容格式为 JSON 数组，特殊字符要求转义，示例：{"value":["\oss-cn-hangzhou.aliyuncs.com\"], "key":"h5_CORSWhiteList"}	空
mp_h5_back_key_ignore_repeat_count	关闭 back 按键的 repeatCount 校验。	YES 表示关闭 repeatCount 校验，其他值表示启用 repeatCount 校验，支持 10.1.32 及以上基线。	NO
mp_h5_allow_mix_content	允许 MixContent 模式，开启此模式有安全风险，请慎重考虑先。 说明： 此开关仅在 10.1.60 版本中支持。	YES 表示允许，NO 表示不允许。	NO

2.6.4 H5 容器扩展

H5Plugin 获取 Activity Result

在如刷脸、识别等许多场景中，都需要启动一个新的 Activity 以获取 Activity 返回的结果，但是在这种场景中，JSAPI 无法直接通过重写 H5Activity 来获取结果。因此在使用 H5 容器时，您需要通过以下方式来获取 Activity 返回的结果：

1. 在自定义的 H5Plugin 中注册 OnH5ActivityResult 回调，示例如下：

```
H5ActivityResultManager.getInstance().put(onH5ActivityResult);
```

说明：

- put 方法不检查重复注册，开发者需自己处理防止重复注册。
- 使用后，需要调用 remove 方法移除回调，一般建议在 H5Plugin 的 onRelease 方法中移除回调，示例如下：

```
H5ActivityResultManager.getInstance().remove(onH5ActivityResult);
```

2. 用 startActivityForResult 的方式启动目的 Activity，如可以在自定义的 H5Plugin 的 handleEvent 方法中启动，示例如下：

```
public boolean handleEvent(H5Event event, H5BridgeContext context) {
    if ("CustomJSAPI".equals(event.getAction())) {
        if (event.getActivity() != null) {
            Intent intent = new Intent(event.getActivity(), yourDestinationActivity.class);
            event.getActivity().startActivityForResult(intent, requestCode, bundle);
        }
        return true;
    }
    return false;
}
```

说明：仅对 H5Activity 的 result 进行 callback 回调。

3. 在 OnH5ActivityResult 的回调方法中，将结果通过 H5BridgeContext 对象传递给前端。

```
public interface OnH5ActivityResult {  
    void onGetResult(int requestCode, int resultCode, Intent intent);  
}
```

自定义 H5 错误页

当您需要自定义 H5 错误页的时候，请按照以下步骤进行操作。

新建一个 html 格式的自定义错误页。

```
<!doctype html>  
<html lang="zh-cn" >  
  
<head>  
<meta charset="utf-8"/>  
<meta name="viewport" content="width=device-width,maximum-scale=1.0,minimum-scale=1.0,user-scalable=no"/>  
<meta name="format-detection" content="telephone=no"/>  
<title>自定义错误</title>  
</head>  
  
<body>  
<p>这个页面是一个自定义错误页</p>  
</body>  
  
</html>
```

2. 实现 H5ErrorPageView。在 APWebView 中设置刚才创建的错误页。

```
public class H5ErrorPageViewImpl implements H5ErrorPageView {  
    @Override  
    public boolean enableShowErrorPage() {  
        // true 表示启动自定义错误页  
        return true;  
    }  
    @Override  
    public void errorPageCallback(H5Page h5Page, APWebView view, String errorUrl, int statusCode, String  
errorMsg, String subErrorMsg) {  
        // 获取错误页的 html，demo 中放到了 raw 中，也可以放在其他地方  
        String html = H5ResourceManager.readRawFromResource(R.raw.custom_error,  
        LauncherApplicationAgent.getInstance().getApplicationContext().getResources());  
        // 将错误页设置给 webview  
        view.loadDataWithBaseURL(errorUrl, html,"text/html","utf-8", errorUrl);  
    }  
}
```

3. 注册 H5ErrorPageView。在打开 H5 容器之前，将自定义的 H5ErrorPageView 注册给容器。

```
H5Utils.setProvider(H5ErrorPageView.class.getName(),new H5ErrorPageViewImpl());
```

开启沉浸式状态栏

说明：此功能仅在 10.1.60 及以上基线版本中支持。

1. 在 H5 容器配置 中开启 TSBS。

对于使用内置标题栏的开发者，可实现 H5TransStatusBarColorProvider 接口，并通过 H5Utils.setProvider 方法将实例设置给 H5 容器，代码示例如下：

```
package com.mpaas.demo.nebula;

import android.graphics.Color;

import com.alipay.mobile.nebula.provider.H5TransStatusBarColorProvider;

public class H5TransStatusBarColorProviderImpl implements H5TransStatusBarColorProvider {
    @Override
    public int getColor() {
        return Color.argb(70, 255, 255, 255);
    }
}
```

说明：

- 此方法将设置所有 H5 容器打开的 H5 页面的状态栏颜色，如果对状态栏颜色设置有更复杂的需求，可实现 H5 容器 自定义标题栏。
- 状态栏颜色设置可在容器标题栏接口的 openTranslucentStatusBarSupport 方法中处理，也可在其他地方自行处理。

添加第三方 JavaScriptInterface

接入方通常会遇到接入第三方页面必须要使用 JavaScriptInterface 的问题，可按照以下步骤来支持此场景：

1. 实现插件以拦截三方页面加载事件。
2. 获取 WebView 并注入 JavaScript 对象。

代码示例如下：

```
package com.mpaas.demo.nebula;

import android.text.TextUtils;

import com.alibaba.fastjson.JSONObject;
import com.alipay.mobile.h5container.api.H5BridgeContext;
import com.alipay.mobile.h5container.api.H5Event;
import com.alipay.mobile.h5container.api.H5EventFilter;
import com.alipay.mobile.h5container.api.H5Param;
import com.alipay.mobile.h5container.api.H5SimplePlugin;
```

```
public class TechFinSitePlugin extends H5SimplePlugin {

    @Override
    public void onPrepare(H5EventFilter filter) {
        super.onPrepare(filter);
        filter.addAction(CommonEvents.H5_PAGE_SHOULD_LOAD_URL);
    }

    @Override
    public boolean interceptEvent(H5Event event, H5BridgeContext context) {
        String action = event.getAction();
        if (CommonEvents.H5_PAGE_SHOULD_LOAD_URL.equals(action)) {
            JSONObject params = event.getParam();
            String url = params.getString(H5Param.LONG_URL);
            if (!TextUtils.isEmpty(url) && url.contains("tech.antfin.com")) {
                event.getH5page().getWebView().addJavascriptInterface(new TechFinJavaScriptInterface(), "techFinBridge");
            }
        }

        return false;
    }
}
```

说明：切勿在 interceptEvent 方法中返回 true ，否则将影响容器加载页面。

```
package com.mpaas.demo.nebula;

import android.webkit.JavascriptInterface;

public class TechFinJavaScriptInterface {

    @JavascriptInterface
    @com.uc.webview.export.JavascriptInterface
    public String whoAmI() {
        return "It is tech fin.";
    }
}
```

说明：系统内核和 UC 内核使用的注解类不同，必须要兼容这两个注解类。

2.6.5 管理 H5 页面

打开 H5 离线包后，您可以对 H5 页面进行以下管理操作：

- 启动全屏的容器传参
- 将单个容器的视图嵌入到页面

启动全屏的容器传参

```
Bundle param = new Bundle();
// 要打开的离线包 URL /www/index.html，必须加 /
// 如果不传 URL，容器将默认打开离线包配置的 URL，建议不要传 URL，只传 appId，前端就可以自己配置打开的页面
param.putString(H5Param.LONG_URL,url);
```

```

LauncherApplicationAgent.getInstance().
getMicroApplicationContext().startApp(null, appId, param);
// 如果是单独接容器的客户端，才使用 h5Service.startPage 的方式
public static final void openH5(String url) {
if (TextUtils.isEmpty(url)) {
return;
}
H5Service h5Service = LauncherApplicationAgent.getInstance().getMicroApplicationContext()
.findServiceByInterface(H5Service.class.getName());
H5Bundle bundle = new H5Bundle();
Bundle param = new Bundle();
// 要打开的离线包 appId
param.putString(H5Param.APP_ID, appId);
// 要打开的离线包 URL /www/index.html，必须加 /
// 如果不传 URL，容器将默认打开离线包配置的 URL，建议不要传 URL，只传appId，前端可以自己配置打开的页面
param.putString(H5Param.LONG_URL,url);
bundle.setParams(param);
if (h5Service != null) {
h5Service.startPage(AlipayApplication.getInstance().getMicroApplicationContext().findTopRunningApp(),
bundle);
}
}
}

```

将单个容器的视图嵌入到页面

根据您的实际情况选择以下方法将单个容器的视图（view）嵌入到 H5 页面中：

- 同步方法
- 异步方法

提示：使用异步方法不占用主线程，不会影响性能。

同步方法

```

public static final void openH5(String url) {
if (TextUtils.isEmpty(url)) {
return;
}
H5Service h5Service = LauncherApplicationAgent.getInstance().getMicroApplicationContext()
.findServiceByInterface(H5Service.class.getName());
H5Bundle bundle = new H5Bundle();
Bundle param = new Bundle();
// 要打开的离线包 appId
param.putString(H5Param.APP_ID, appId);
// 要打开的离线包内的 URL /www/index.html，必须加 /
// 如果不传 URL，容器将默认打开离线包默认配置的 URL
param.putString(H5Param.LONG_URL,url);
bundle.setParams(param);
if (h5Service != null) {
H5Page h5Page=h5Service.createPage(activity,bundle);
View view=h5Page.getContentView(),
// view 最后添加到自己的页面中就行
}
}
}

```

```
H5Service h5Service = LauncherApplicationAgent.getInstance().getMicroApplicationContext()
    .findServiceByInterface(H5Service.class.getName());
H5Bundle bundle = new H5Bundle();
Bundle param = new Bundle();
param.putString(H5Param.APP_ID, appId);
param.putString(H5Param.LONG_URL, url);
bundle.setParams(param);
if (h5Service != null) {
    h5Service.createPageAsync(activity, bundle, h5PageReadyListener);
}
```

2.7 使用教程

2.7.1 总览

mPaaS Inside 是 mPaaS 提供的新的接入方式。这种新的接入方式与原生工程接入 aar 的方式几乎一致，但可以使用 mPaaS 提供的能力，适合于对 Bundle 组件化方案没有特别需求却急需使用 mPaaS 能力的客户。为了方便您快速熟悉并掌握这种新的接入方式，我们提供了本教程帮助您快速上手。

本教程一共包含以下六个部分：

1. 在 Android Studio 创建应用
2. 在 mPaaS 控制台创建应用
3. 接入 mPaaS Inside 到原生工程
4. 添加 H5 容器组件到工程
5. 使用H5 容器
6. 使用H5 离线包

您将学会

- 如何创建一个通过点击按钮弹出 toast 的安卓应用。
- 如何接入 mPaaS Inside。
- 如何使用 mPaaS H5 容器服务。
- 如何使用 mPaaS H5 离线包服务。

您将需要

1. 配置开发环境（在本教程中以 macOS 下的开发环境为例进行说明）。更多信息，请参考 [准备配置](#)。
2. 网络访问。
3. 网络浏览器（建议您使用 Chrome 浏览器）。
4. 一部安卓手机及配套的数据线，系统版本为安卓 4.3 或更新（您也可以选择使用模拟器进行调试，在本教程中以实际手机为例）。

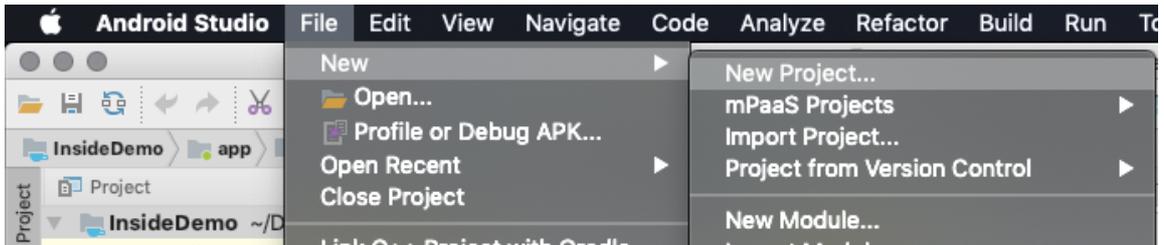
2.7.2 在 Android Studio 创建应用

在本节您将创建一个通过点击按钮弹出 toast 的应用，并获得 APK 格式的安装包。该过程主要分为五个步骤：**创建工程**、**编写代码**、**创建签名文件**、**给工程添加签名**、和**在手机上安装应用**。

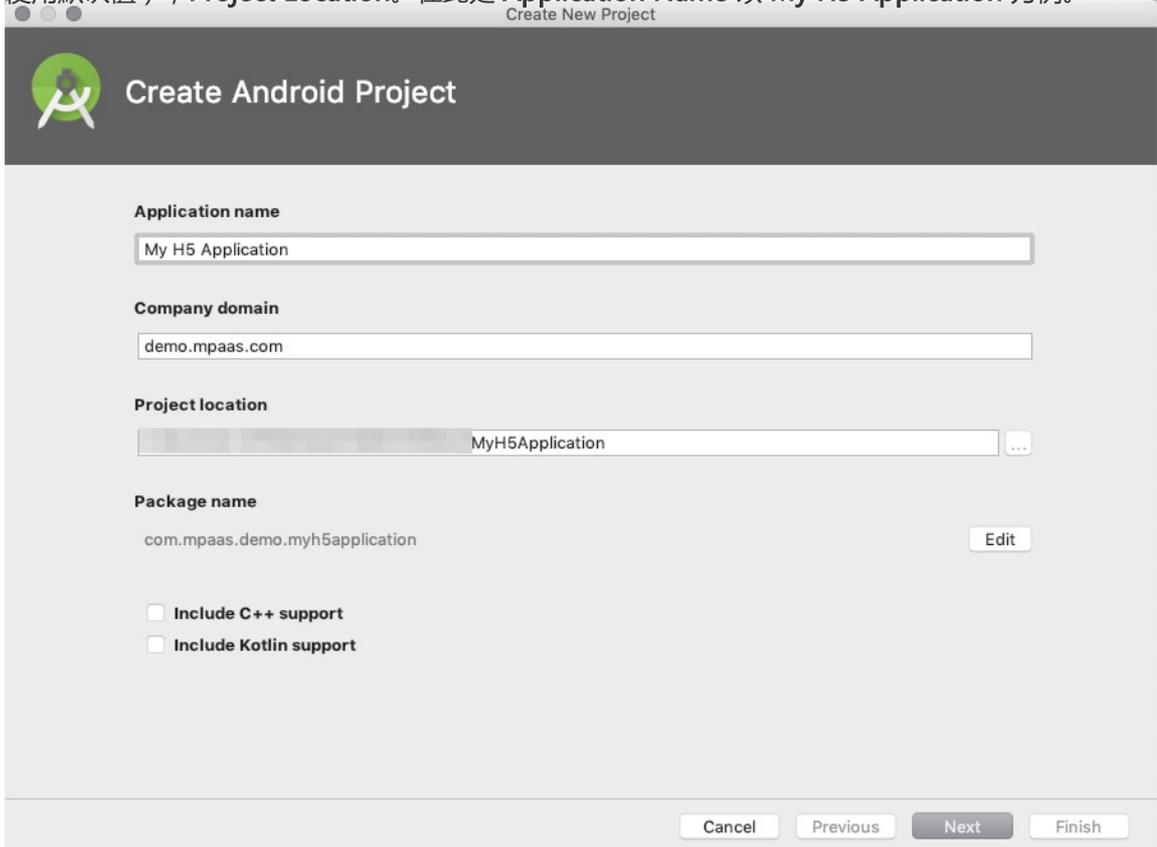
如果您已经有了一个原生的 Android 开发工程并完成了签名，那么您可以直接跳转到在 mPaaS 控制台创建应用。

创建工程

打开 Android Studio。

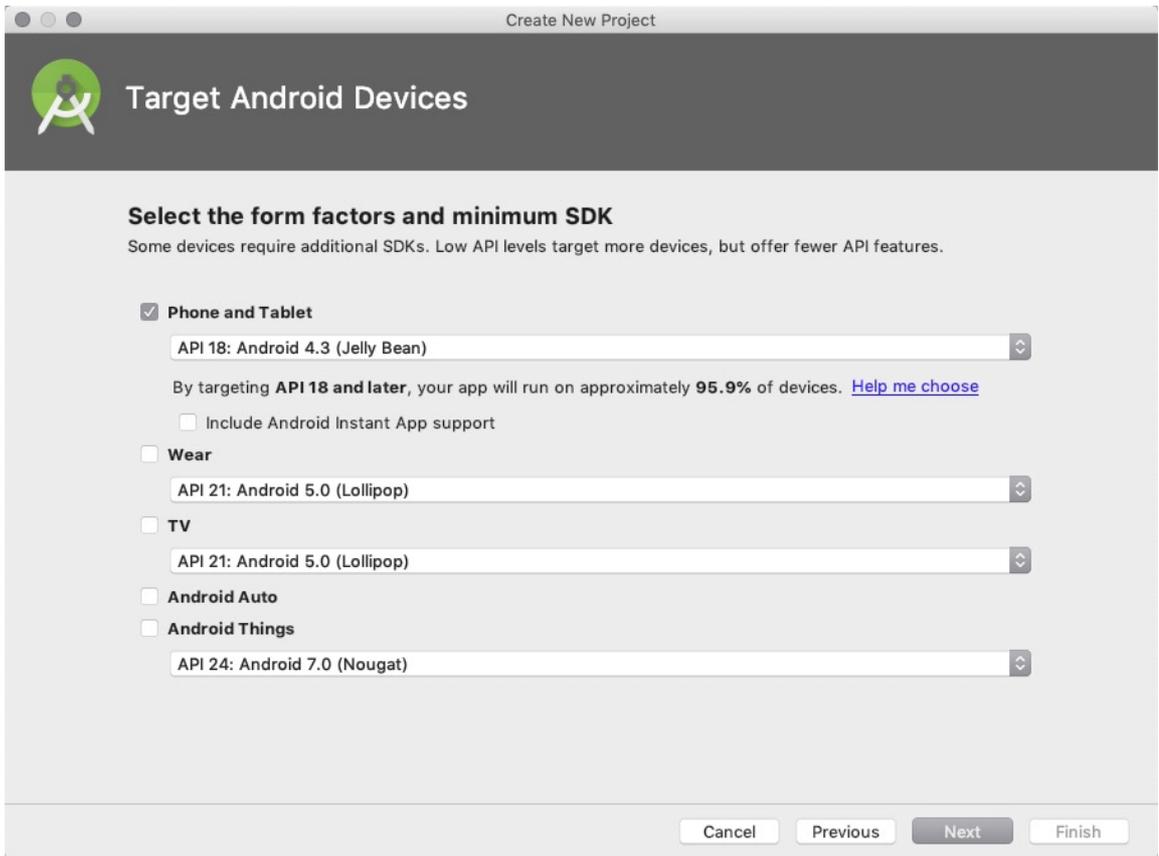


点击 **File > New > New Project**，输入 **Application Name**，**company domain**（如有；否则可以使用默认值），**Project Location**。在此处 **Application Name** 以 **My H5 Application** 为例。

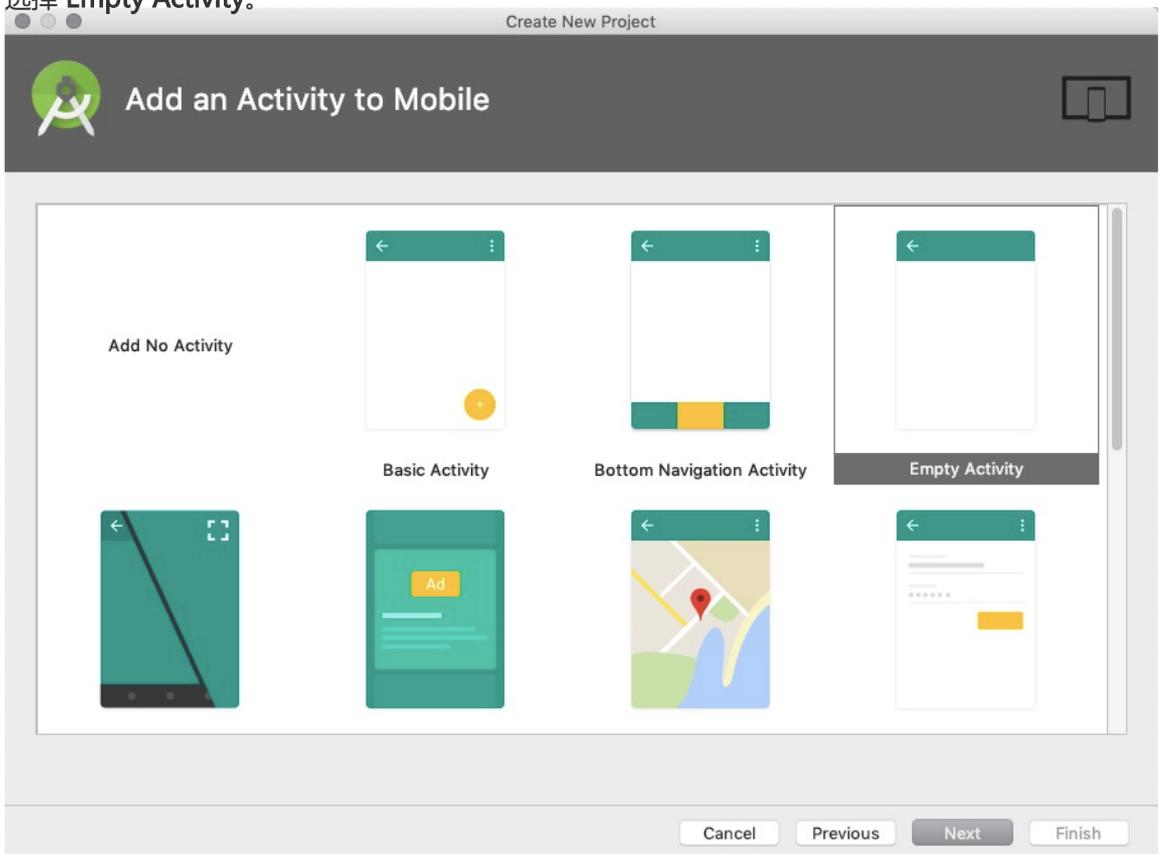


选择 **Target Android Devices** 为 **Phone and Tablet**，并选择 **API Target level** 为 **API 18: Android 4.3 (Jelly Bean)**。

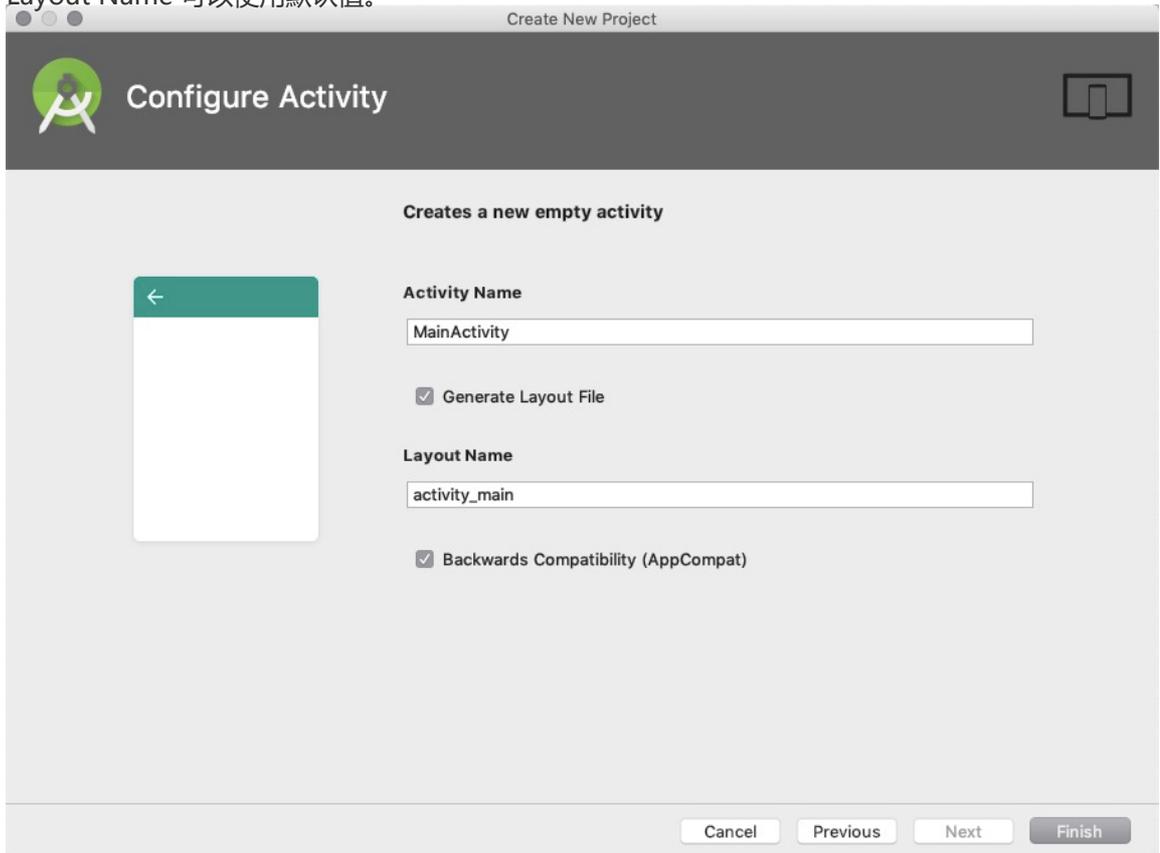
说明：API 18: Android 4.3 (Jelly Bean) 是 mPaaS 及 mPaaS Inside 支持的最低版本，您在实际生产中可以根据需要进行选择。



选择 Empty Activity.



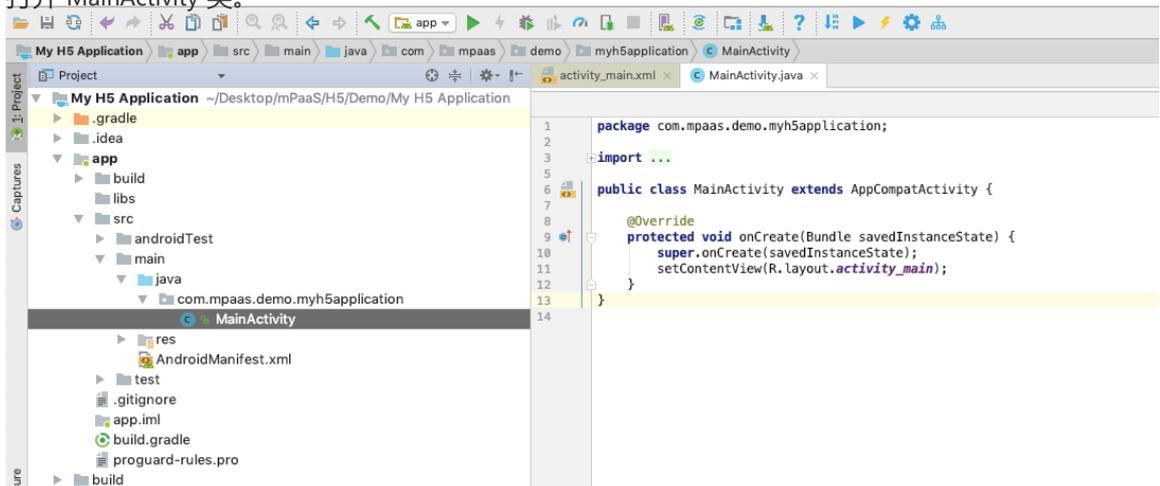
勾选 **Generate Layout File** 和 **Backwards Compatibility (AppCompat)**。Activity Name 和 Layout Name 可以使用默认值。



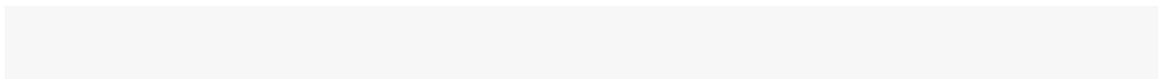
点击 **Finish**。至此您已完成 **创建工程**。

编写代码

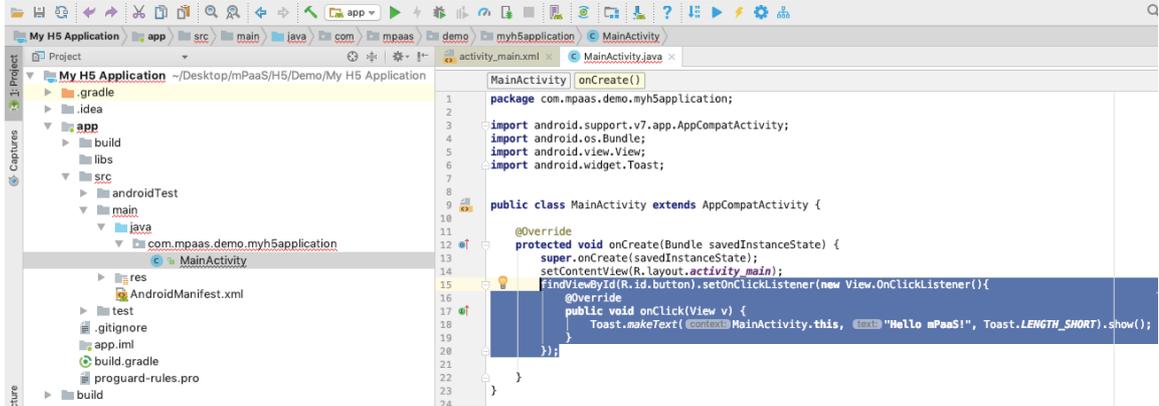
1. 打开 MainActivity 类。



输入以下代码。

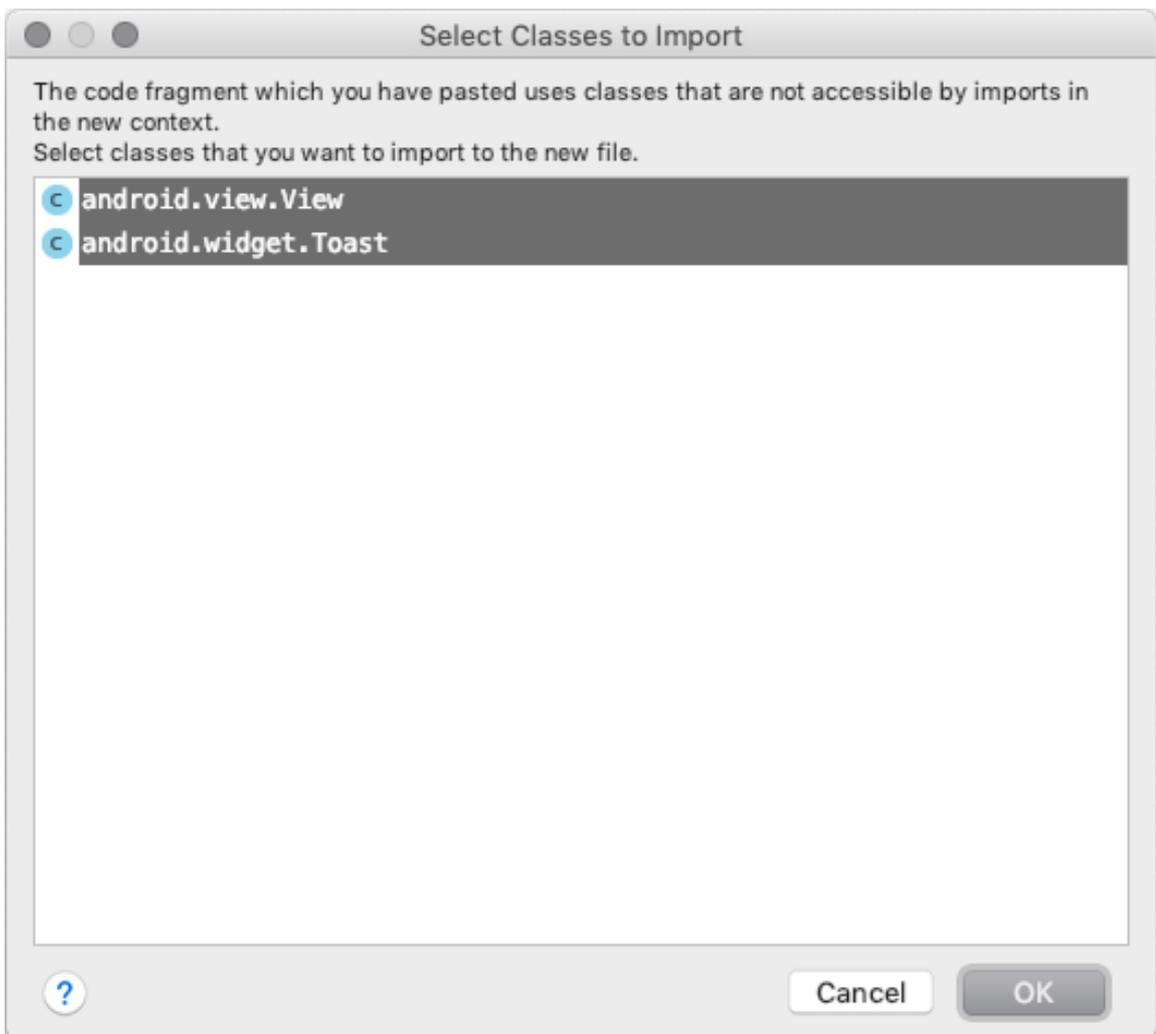


```
findViewById(R.id.button).setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v) {
        Toast.makeText(MainActivity.this, "Hello mPaaS!", Toast.LENGTH_SHORT).show();
    }
});
```

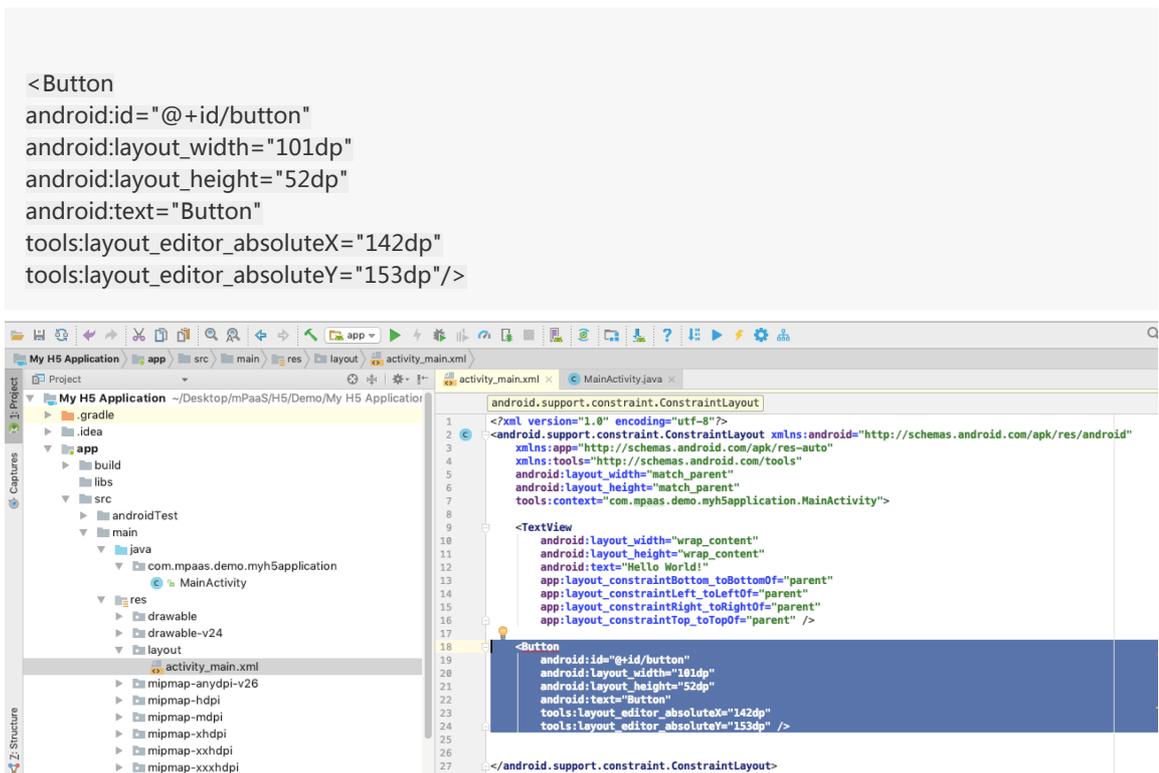


The screenshot shows the Android Studio IDE. The left sidebar displays the project structure for 'My H5 Application'. The main editor window shows the code for MainActivity.java. The code includes imports for AppCompatActivity, Bundle, View, and Toast. The MainActivity class extends AppCompatActivity and has an onCreate method and an onClick method. A lightbulb icon is shown next to the findViewById and Toast.makeText calls, indicating that the classes View and Toast are not imported.

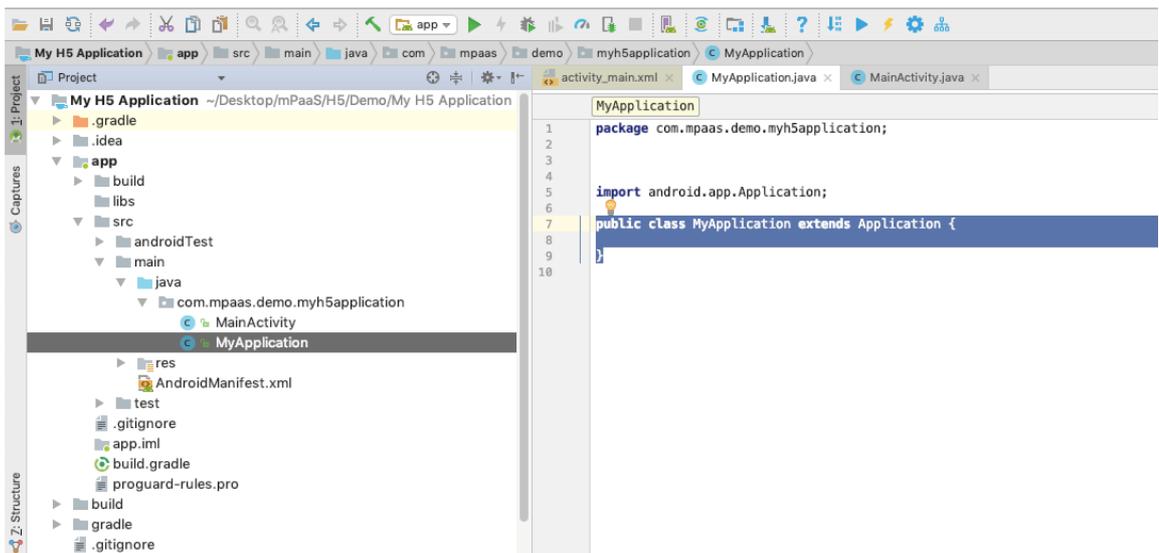
说明：如果您粘贴以上代码时 Android Studio 弹出了以下窗口，请点击 OK。



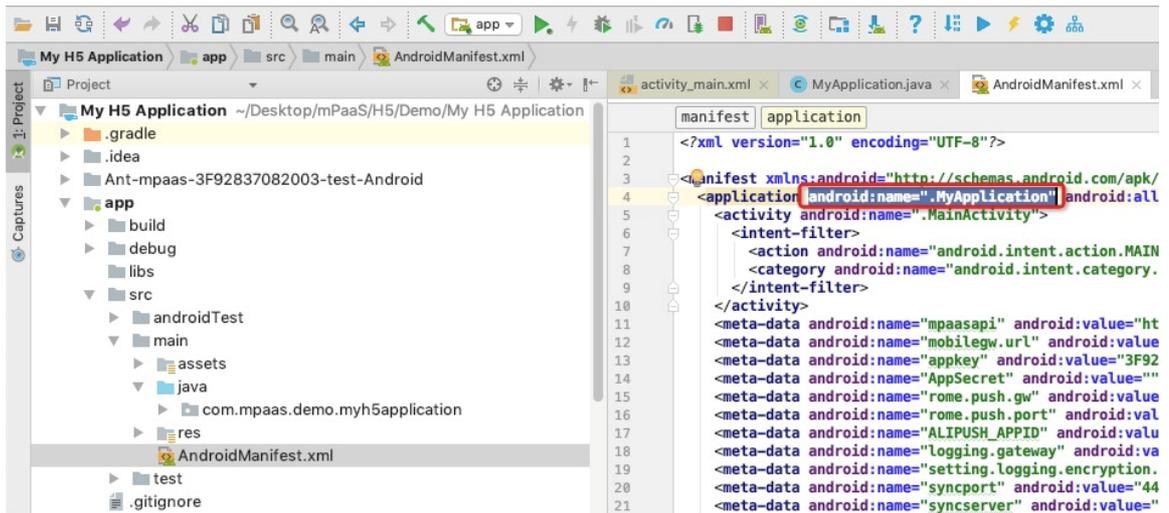
打开 activity_main.xml 文件，输入以下代码。



给工程添加自定义的类 MyApplication，该类继承自 Application。



打开 AndroidManifest.xml，添加 android:name=".MyApplication"到 <application> 标签。

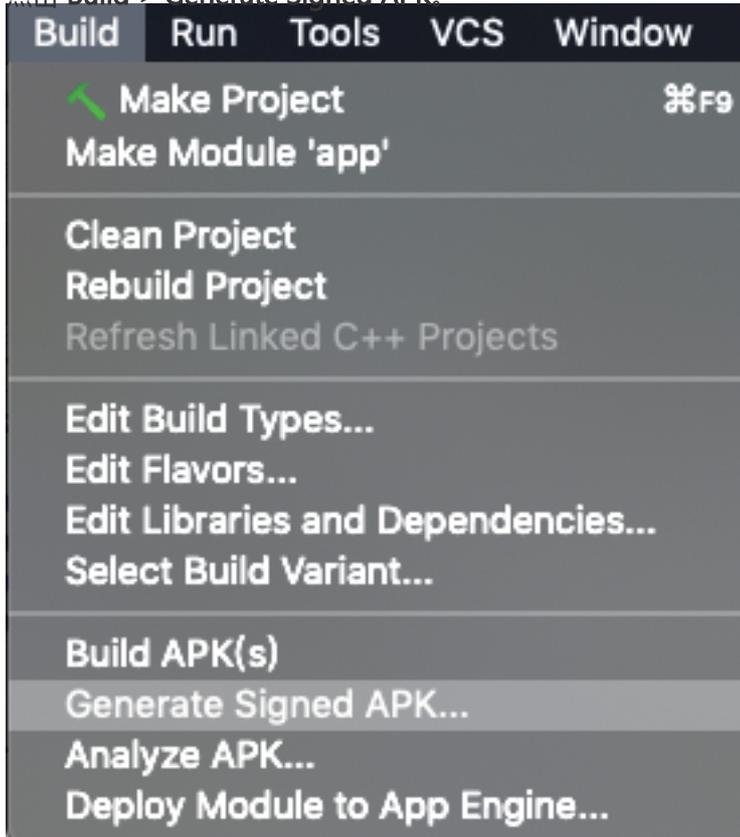


编译并运行。至成功运行，您已完成 **编写代码**。

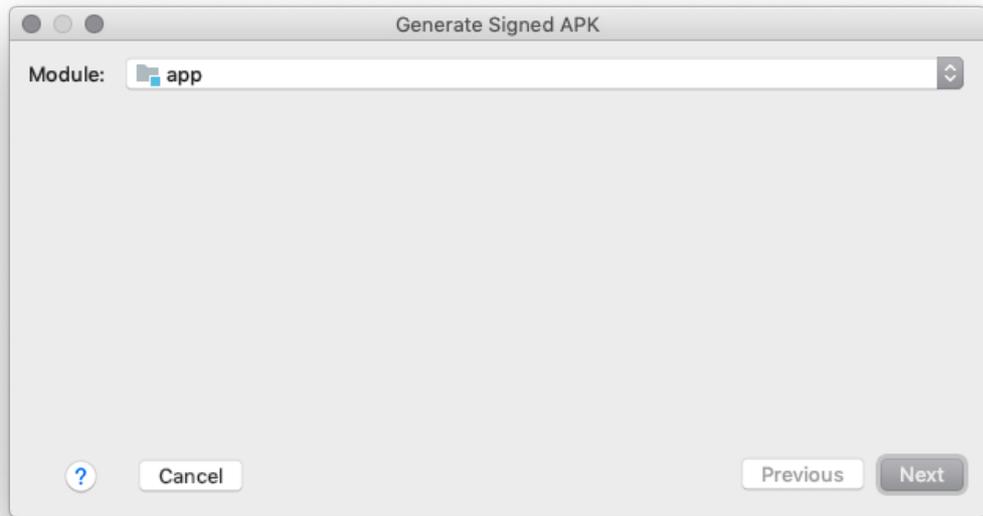
创建签名文件

如果您已经拥有签名文件，可直接跳转至 **给工程添加签名**。

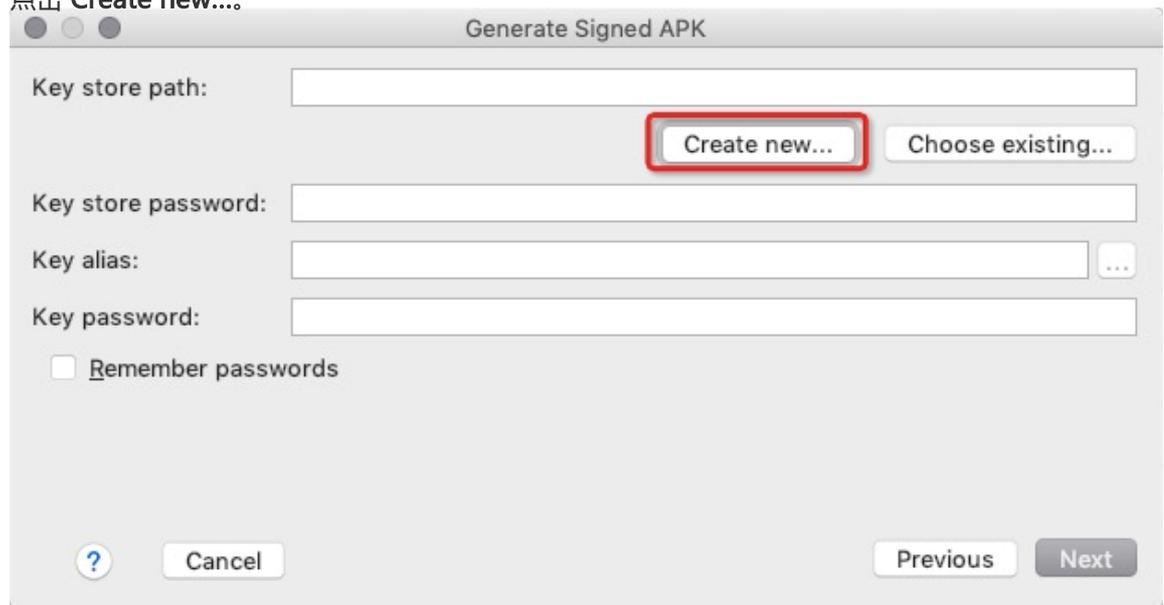
点击 **Build > Generate Signed APK**。



点击 **Next**。



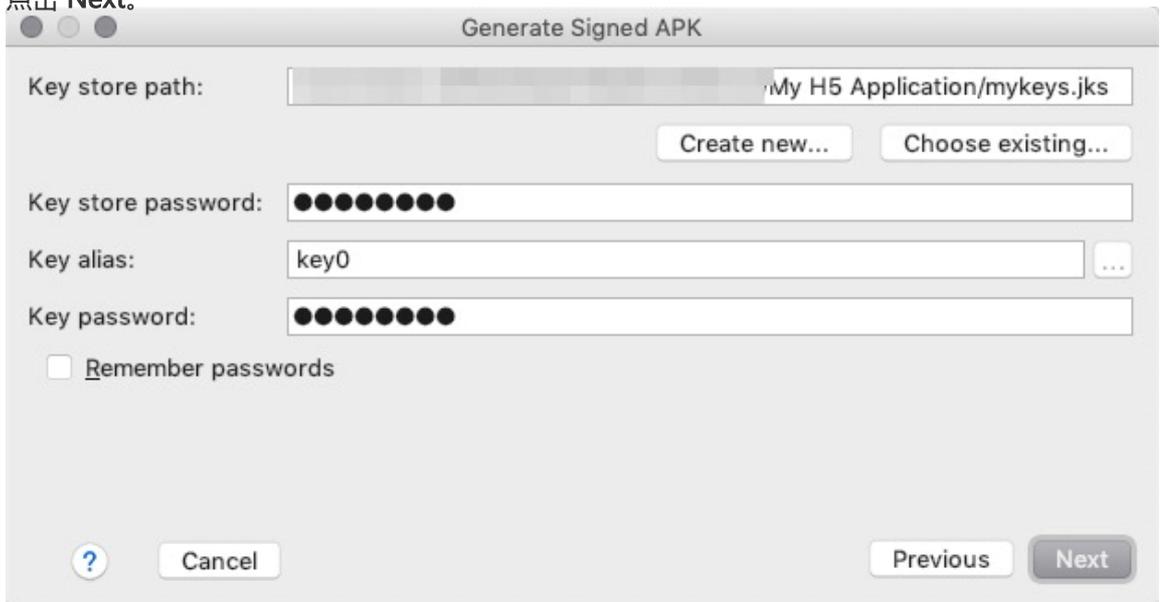
点击 Create new...



输入相应内容后，点击 OK。

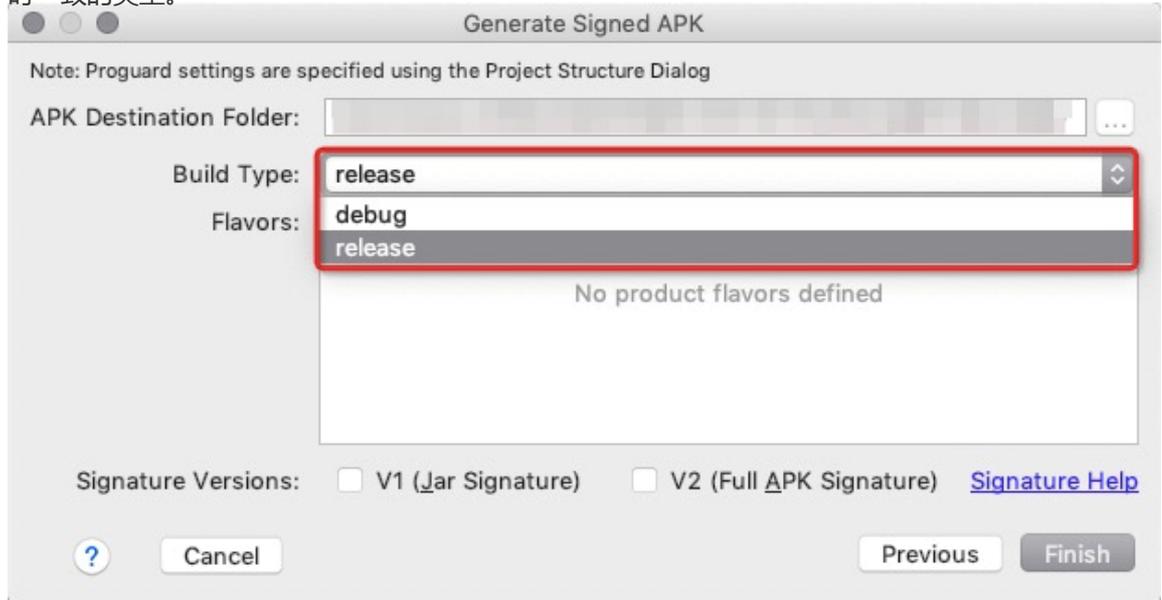


点击 Next。



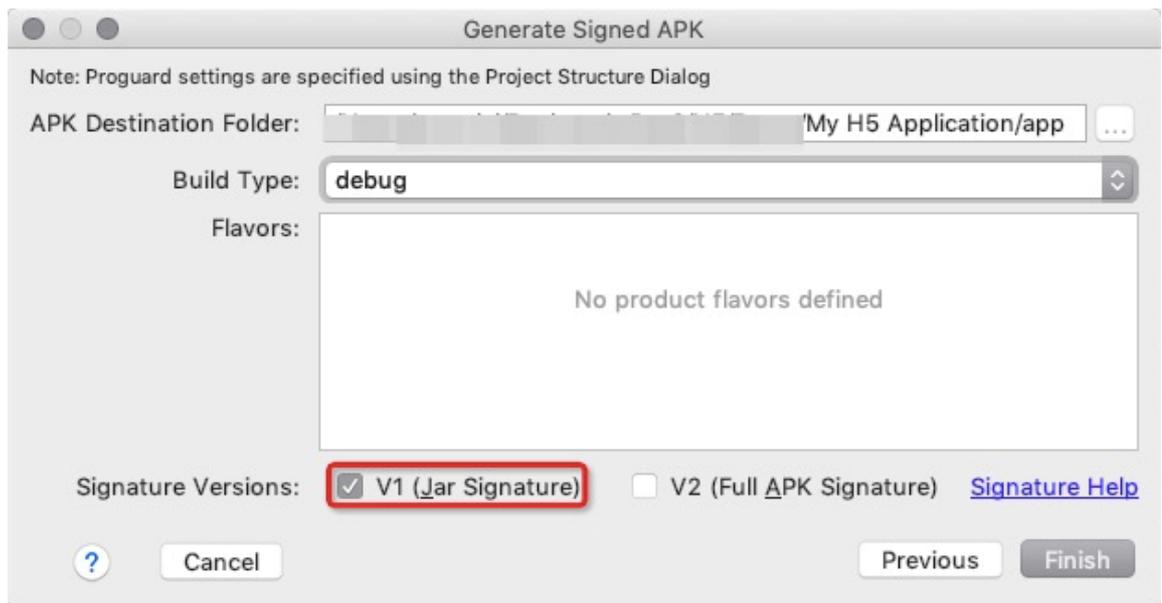
根据需要选择 **Build Type**。Build Type 信息需要牢记，因为在使用加密文件的时候需要选择和生成

时一致的类型。



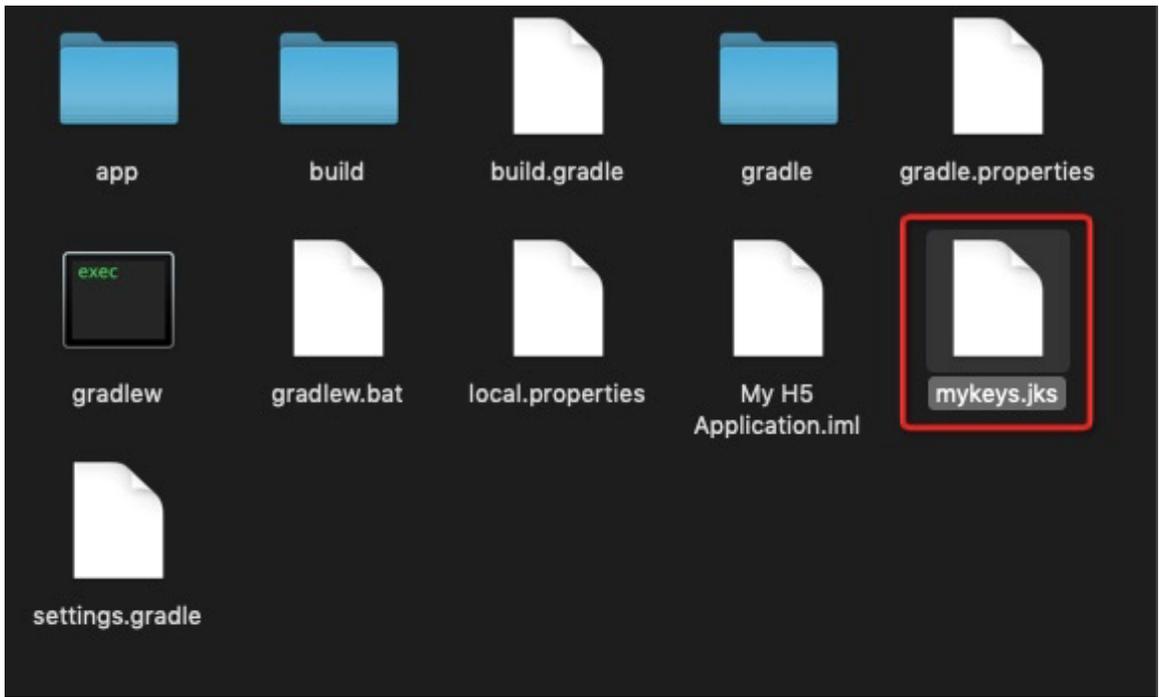
勾选 **V1 (Jar Signature)** 加密版本。

V1 (Jar Signature) 为必选项，**V2 (Full APK Signature)** 可按需选择。



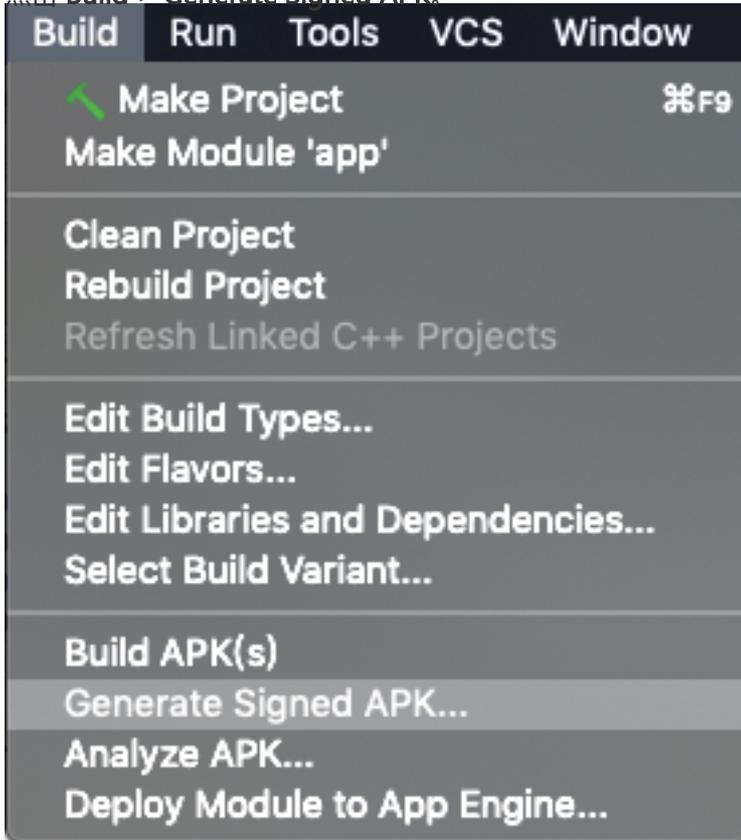
点击 **Finish**。片刻后，在指定的路径下即可获得生成的签名文件。至此，您已完成 **创建签名文件**。在工程文件夹下的 **debug** 文件夹（`/My H5 Application/app/build/outputs/apk/debug`）中，即可获得该应用签名后的 **APK 安装包**。

说明：在生成新的签名文件后，Android Studio 会自动编译并运行工程。

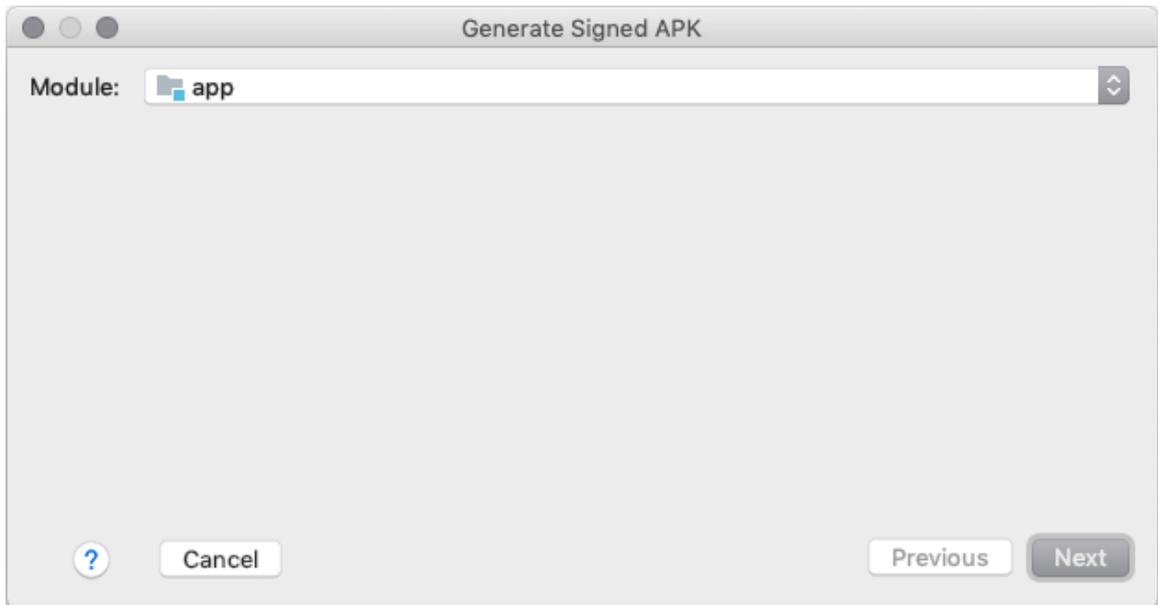


给工程添加签名

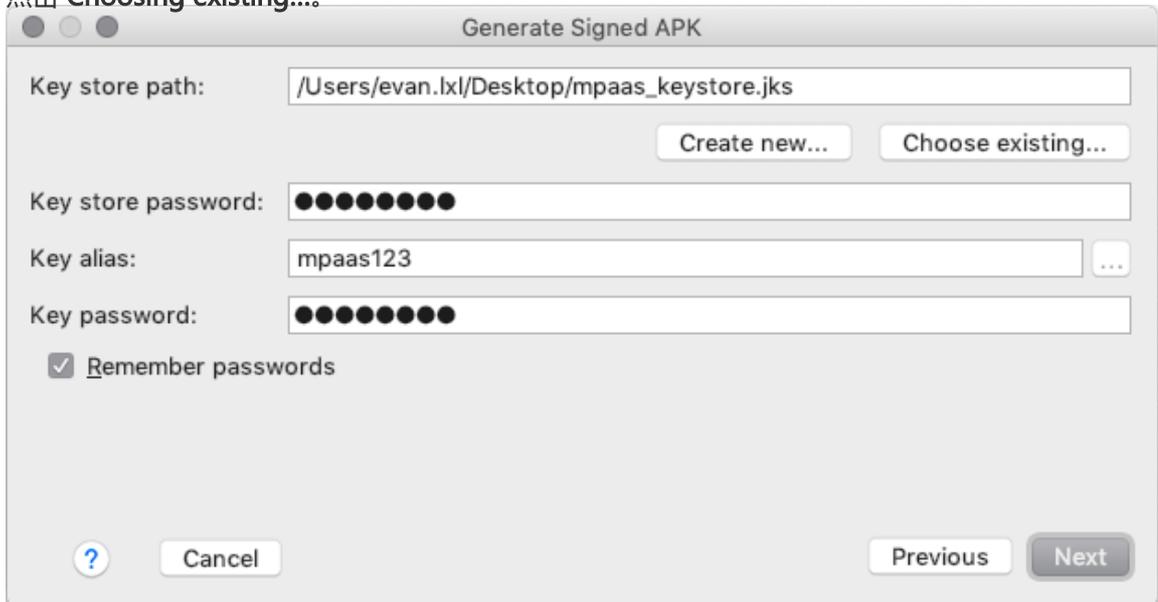
点击 Build > Generate Signed APK.



点击 Next.



点击 **Choosing existing...**



指定到您本地的签名文件，并输入 **Key store password**、**Key alias** 和 **Key password**，点击 **Next**。

5. 在 **Build Type** 中选择您生成签名文件时选择的类型，并根据需要勾选 **Signature Versions**。点击 **Finish**。至此，您已完成 **给工程添加签名**。在工程文件夹下的 debug 文件夹（/My H5 Application/app/build/outputs/apk/debug）中，即可获得该应用签名后的APK安装包。

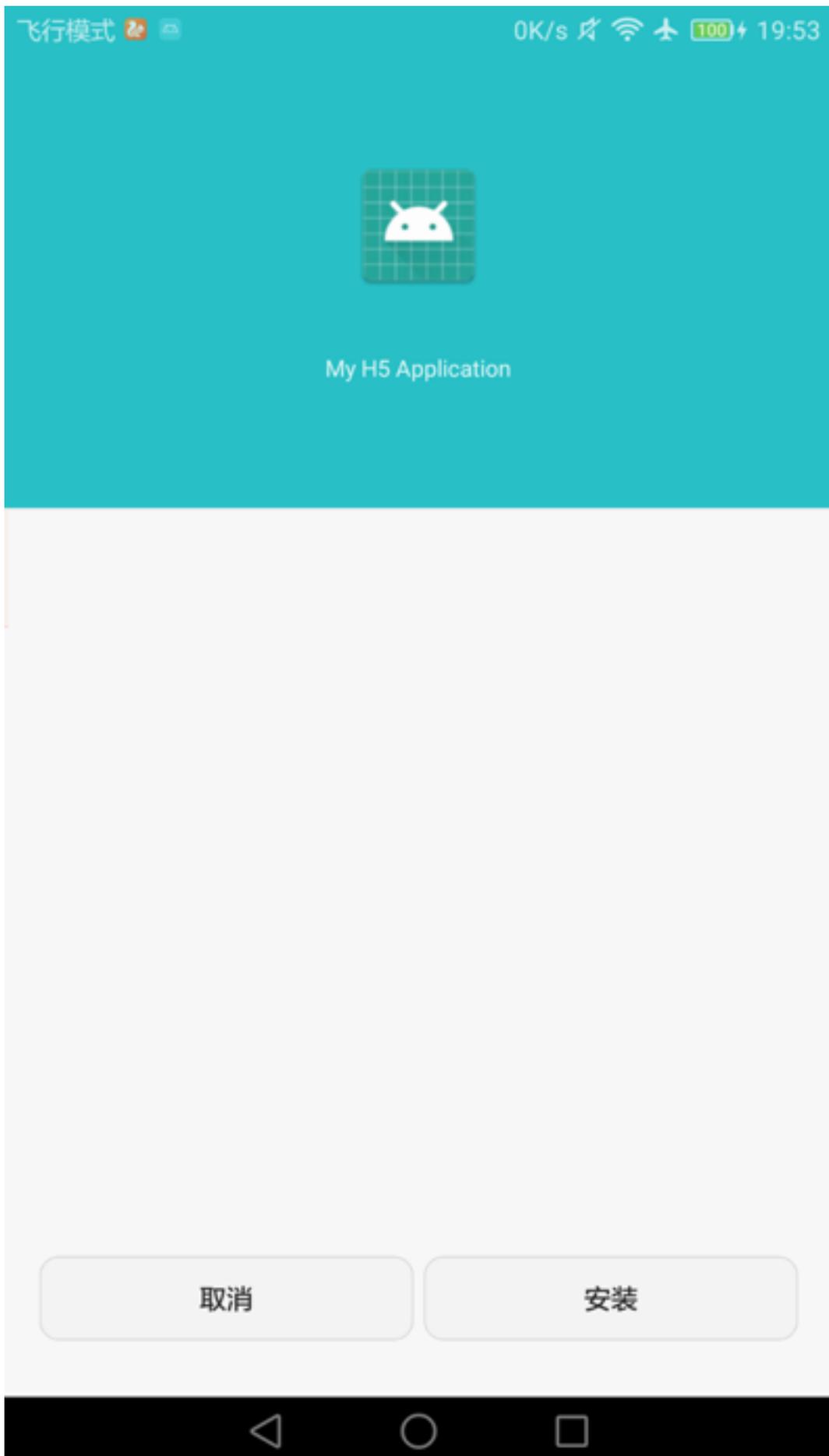
说明：在添加签名文件后，Android Studio 会自动编译并运行工程。

在手机上安装应用

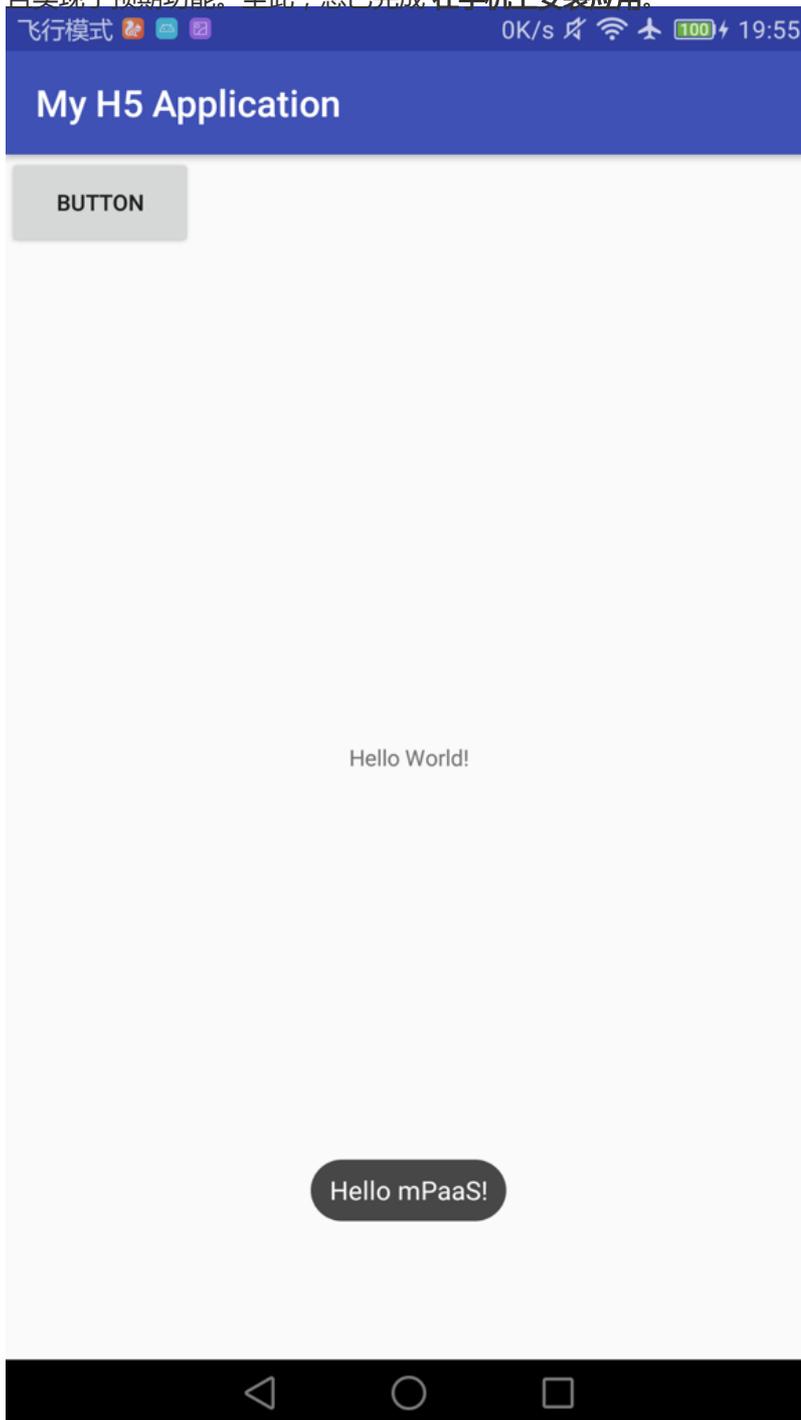
1. 连接手机到电脑，并开启手机的 USB 调试模式。
2. 运行工程并在弹出的 **Select Deployment Target** 窗口中选择您的手机，点击 **OK**。



3. 在手机上点击 **安装** 以安装应用。



4. 在手机上打开应用，点击 **BUTTON**，弹出如图所示 Toast “Hello mPaaS!”，即表示应用安装成功且实现了预期功能。至此，您已完成 **在手机上安装应用**。



2.7.3 在 mPaaS 控制台创建应用

1. 打开网络浏览器，登录 [mPaaS 控制台](#)。
2. 创建 mPaaS 应用。

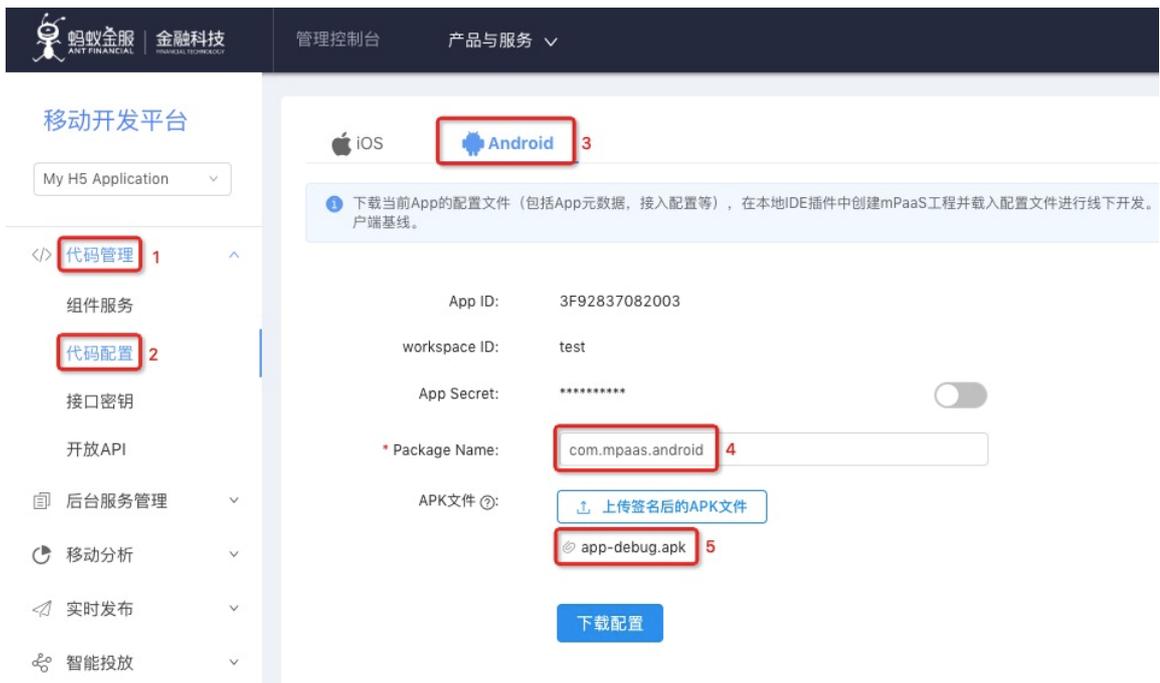


3. 输入项目名并点击 **确定**。



4. 点击刚创建的应用，进入应用。

点击 **代码管理** > **代码配置** > **Android**，输入 **Package Name**（此处以 `com.mpaas.android` 为例），上传编译并加签后的 APK 安装包。最后，点击 **下载配置** 下载配置文件。



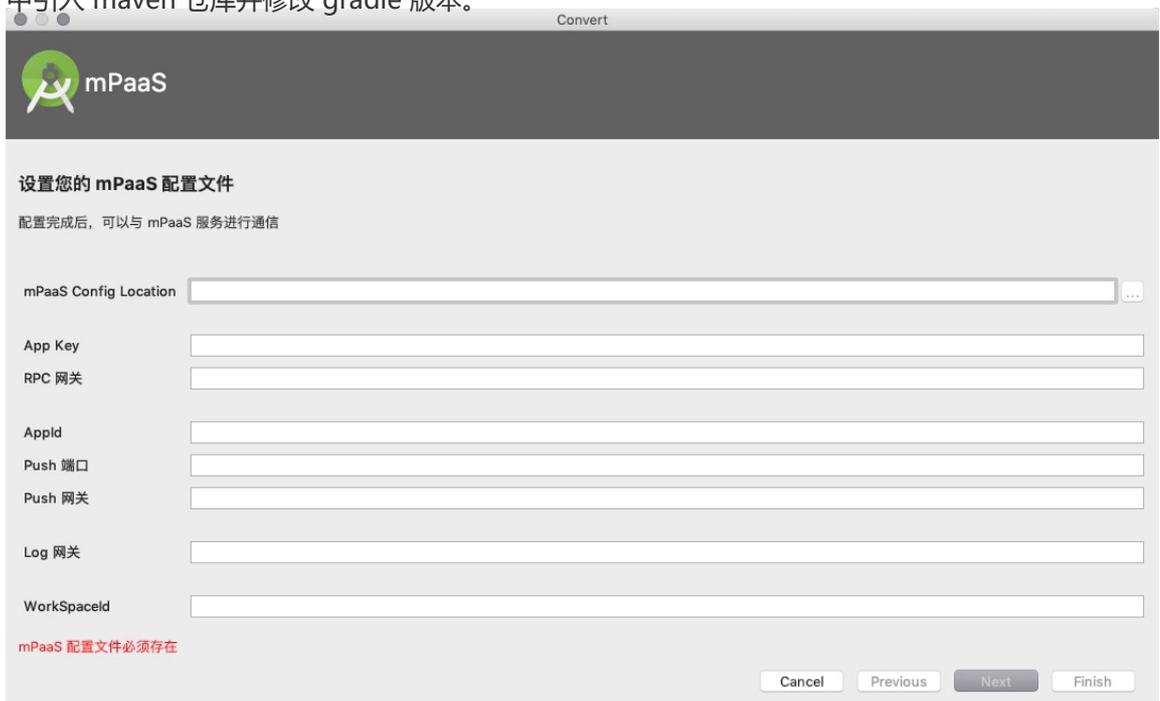
5. 下载到的配置文件是一个压缩包，解压该压缩包，会得到一个配置文件和一个加密图片。

2.7.4 接入mPaaS Inside 到工程

在客户端接入 mPaaS Inside 到 My H5 Application 工程。

点击 [查看视频](#)，查看与本教程对应的视频教程。

在 **mPaaS > 安装 mPaaS Inside** 菜单下的 **mPaaS 工程迁移** 页面，选择在控制台下载到的配置文件，Android Studio 会自动加载配置文件的信息，并根据得到的信息在修改 build.gradle 文件以在工程中引入 maven 仓库并修改 gradle 版本。



您需要注意以下事项：

- 如果您在工程中使用了 module 工程，那么您还需要修改 build.gradle 文件以引入以下额外的插件，这些插件能够帮助您更好地接入 mPaaS 的能力。

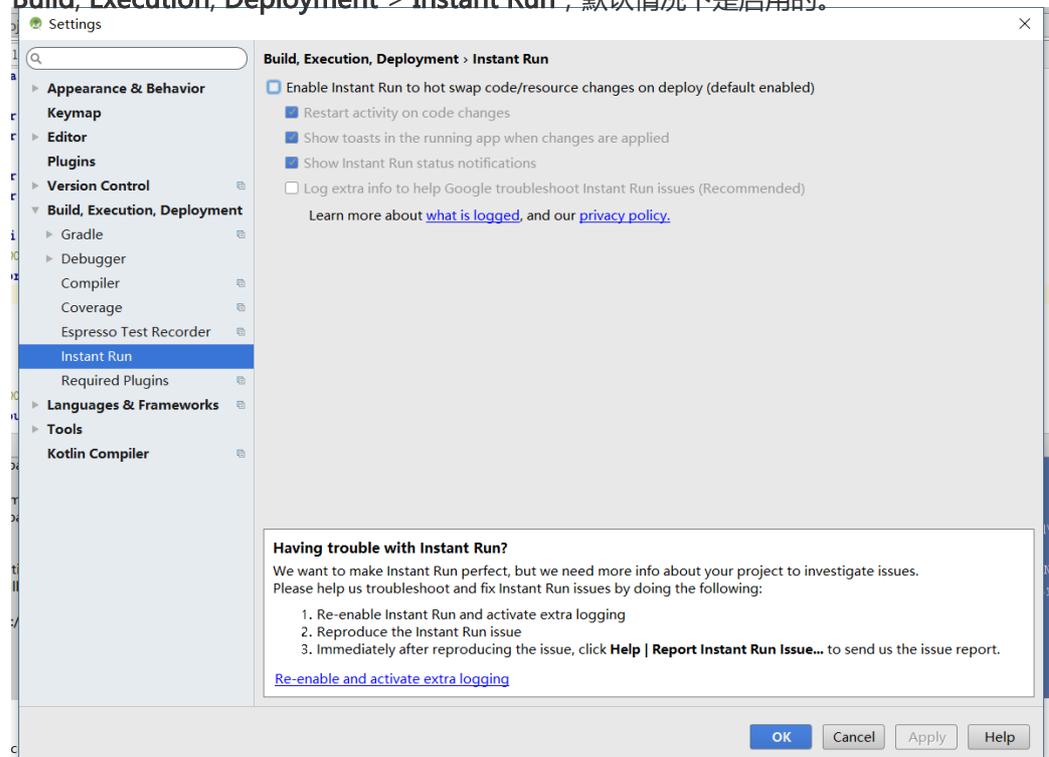
```
apply plugin: 'com.alipay.library'
apply plugin: 'com.alipay.apollo.baseline.update'
```

如果您不需要 mPaaS 提供一些组件，把它们的以字符串 group:artifact 加入到 excludeDependencies 数组中。

```
mpaascomponents{
// 如果您不需要 mPaaS 提供一些组件，把他们的以字符串 group:artifact 加入到下面的数组中
//例 excludeDependencies=[
//例"com.alipay.android.phone.thirdparty:androidsupport-build",
//例"com.alipay.android.phone.thirdparty:androidsupportrecyclerview-build"
//例 ]
excludeDependencies=[

]
}
```

- 接入 mPaaS Inside 后，需要关闭 **Enable Instant Run to hotswap code / resource changes on deploy** 选项，否则会编译失败。该选项在 **Android Studio > Preference > Build, Execution, Deployment > Instant Run**，默认情况下是启用的。



2. 编译并运行工程。如果都能成功，则说明接入 mPaaS Inside 到工程成功。至此，您已经完成 接入

mPaaS Inside 到工程。

2.7.5 添加 H5 组件到工程

点击 [查看视频](#)，查看与本教程对应的视频教程。

1. 点击 mPaaS > 组件管理。



2. 选择 NEBULA - H5 容器 组件，点击 **安装**。安装成功后，按钮显示为 **卸载**。



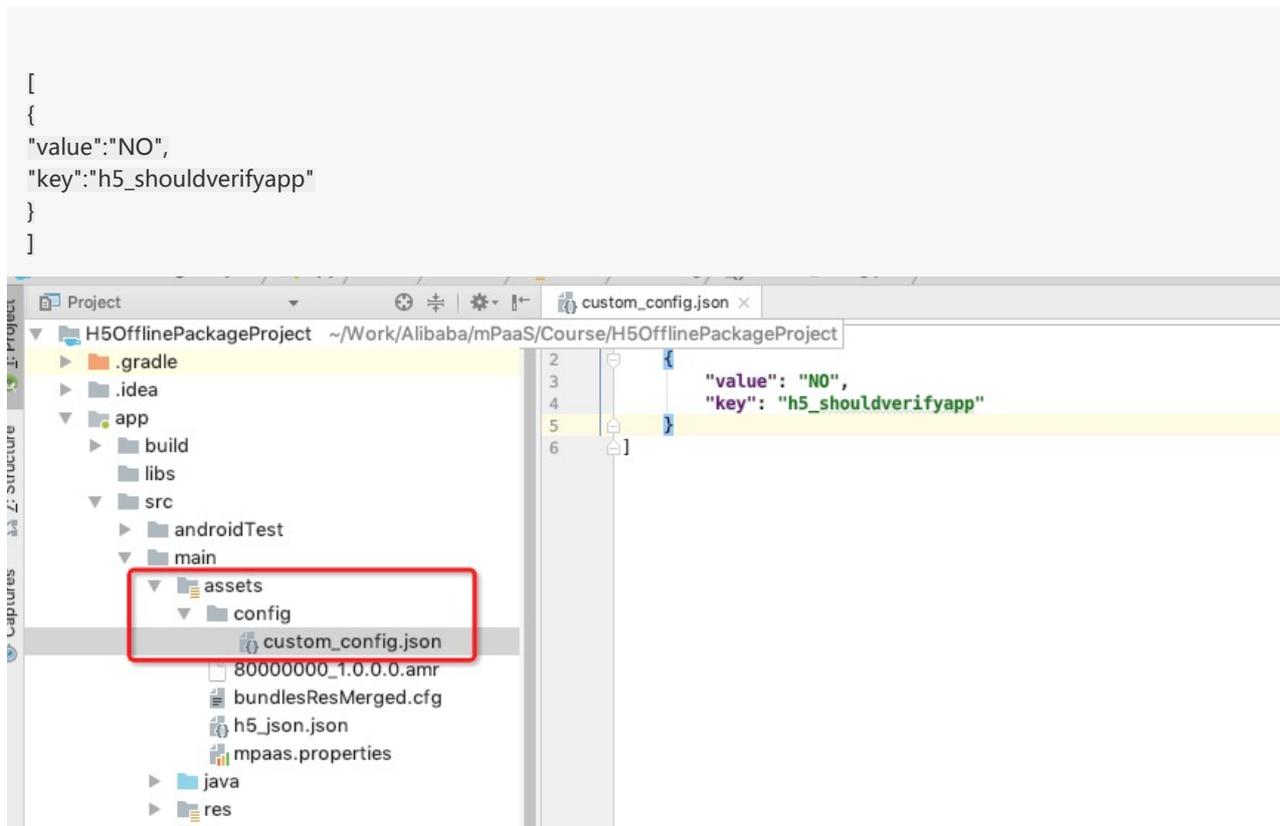
3. 安装完成后，编译工程通过。至此，您已经添加 H5 容器组件到您的工程了。

4. (可选) 关闭 H5 容器验签。

在默认情况下，H5 容器的验签是开启的。如果您需要关闭验签，则需要

在 `/app/src/main/assets/config` 路径下，添加 `custom_config.json` 配置文件，并输入以下代码以关闭验签

。



2.7.6 使用 H5 容器

您可使用 H5 容器完成以下操作：

- 在应用内打开一个在线网页
- 前端调用 Native 接口
- 前端调用自定义 JSAPI
- 自定义H5 页面的 TitleBar
- 使用 UC 内核

在应用内打开一个在线网页

点击 [查看视频](#)，查看对应的视频教程。

初始化 mPaaS Inside。

初始化可以在自定义的 Application 类或其他应用进程中较早初始化的地方调用框架方法进行，此处
在自定义的 MyApplication 类中进行初始化。初始化方法如下：

```
package com.mpaas.demo.myh5application;

import android.app.Application;
import android.content.Context;

import com.alipay.mobile.framework.quinoxless.IInitCallback;
import com.alipay.mobile.framework.quinoxless.QuinoxlessFramework;

public class MyApplication extends Application {
    @Override
    protected void attachBaseContext(Context base) {
        super.attachBaseContext(base);
        QuinoxlessFramework.setup(this, new IInitCallback() {
            @Override
            public void onPostInit() {
                // 在这里开始使用mPaaS功能
            }
        });
    }

    @Override
    public void onCreate() {
        super.onCreate();
        QuinoxlessFramework.init();
    }
}
```

在 app/src/main/AndroidManifest.xml 文件中，添加 android:name=".MyApplication"。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.mpaas.demo.myh5application">

    <application
```

```
android:name=".MyApplication"
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:roundIcon="@mipmap/ic_launcher_round"
android:supportsRtl="true"
android:theme="@style/AppTheme">
<activity android:name=".MainActivity">
<intent-filter>
<action android:name="android.intent.action.MAIN"/>

<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
</application>

</manifest>
```

在 activity_main.xml 文件中，重写之前的按钮 button，将 button id 改为 start_url_btn，并进行简单样式调整。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

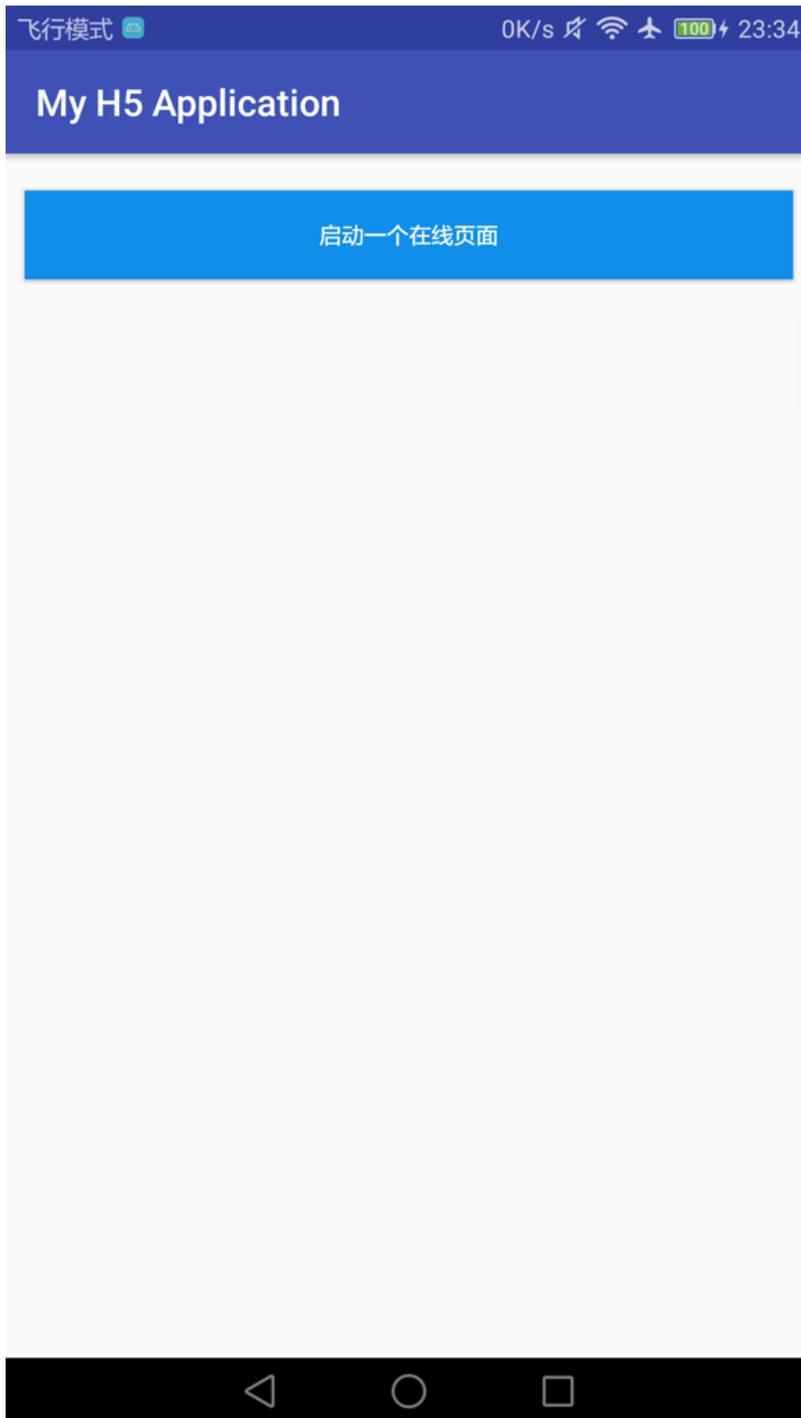
<Button
android:id="@+id/start_url_btn"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginBottom="10dp"
android:layout_marginLeft="10dp"
android:layout_marginRight="10dp"
android:layout_marginTop="20dp"
android:background="#108EE9"
android:gravity="center"
android:text="启动一个在线页面"
android:textColor="#ffffff"/>

</LinearLayout>
```

在 MainActivity 类重写点击按钮后的行为，实现打开 [蚂蚁金服金融科技](https://tech.antfin.com/) 官网的功能。实现代码如下所示：

```
findViewById(R.id.start_url_btn).setOnClickListener(new View.OnClickListener(){
@Override
public void onClick(View v) {
MPNebula.startUrl("https://tech.antfin.com/");
}
});
```

编译工程后，在手机上安装应用。打开应用后界面如下：



6. 点击按钮后即可应用内打开金融科技官网首页，即说明接口调用成功。至此，您已完成 **在应用内打开一个在线网页**。



前端调用 Native 接口

点击 [查看视频](#)，查看对应的视频教程。

在开发前端页面时，可以通过 Nebula 容器提供的 bridge，通过 JSAPI 的方式与 Native 进行通信，获取 Native 处理的相关信息或数据。Nebula 容器内预置了部分基础的 JSAPI 能力（详情请查看[链接](#)），您可以在 H5 页面的 js 文件中，直接通过 AlipayJSBridge.call 的方式进行调用。示例如下：

```
AlipayJSBridge.call('alert', {
```

```
title: '原生 Alert Dialog',  
message: '这是一个来自原生的 Alert Dialog',  
button: '确定'  
}, function (e) {  
alert("点击了按钮");  
});
```

说明： https://mcube-prod.oss-cn-hangzhou.aliyuncs.com/570DA89281533-default/80000000/1.0.0.1_all/nebula/fallback/h5_to_native.html 是已经写好的前端页面，您可以调用此页面以体验前端调用Native接口的功能。

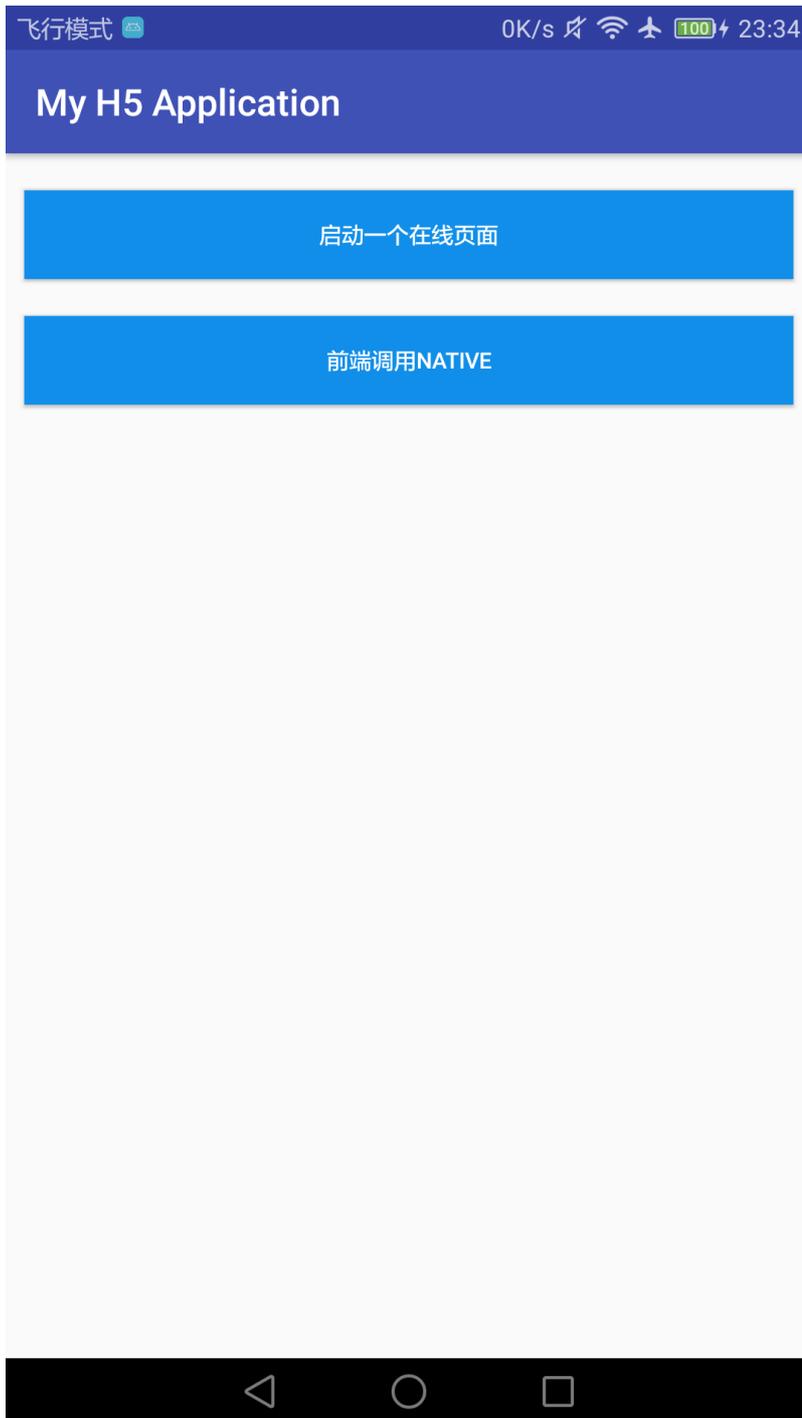
在 activity_main.xml 文件中，新增按钮 button， button id 改为 h5_to_native_btn。

```
<Button  
android:id="@+id/h5_to_native_btn"  
android:layout_width="match_parent"  
android:layout_height="wrap_content"  
android:layout_margin="10dp"  
android:background="#108EE9"  
android:gravity="center"  
android:text="前端调用Native"  
android:textColor="#ffffff"/>
```

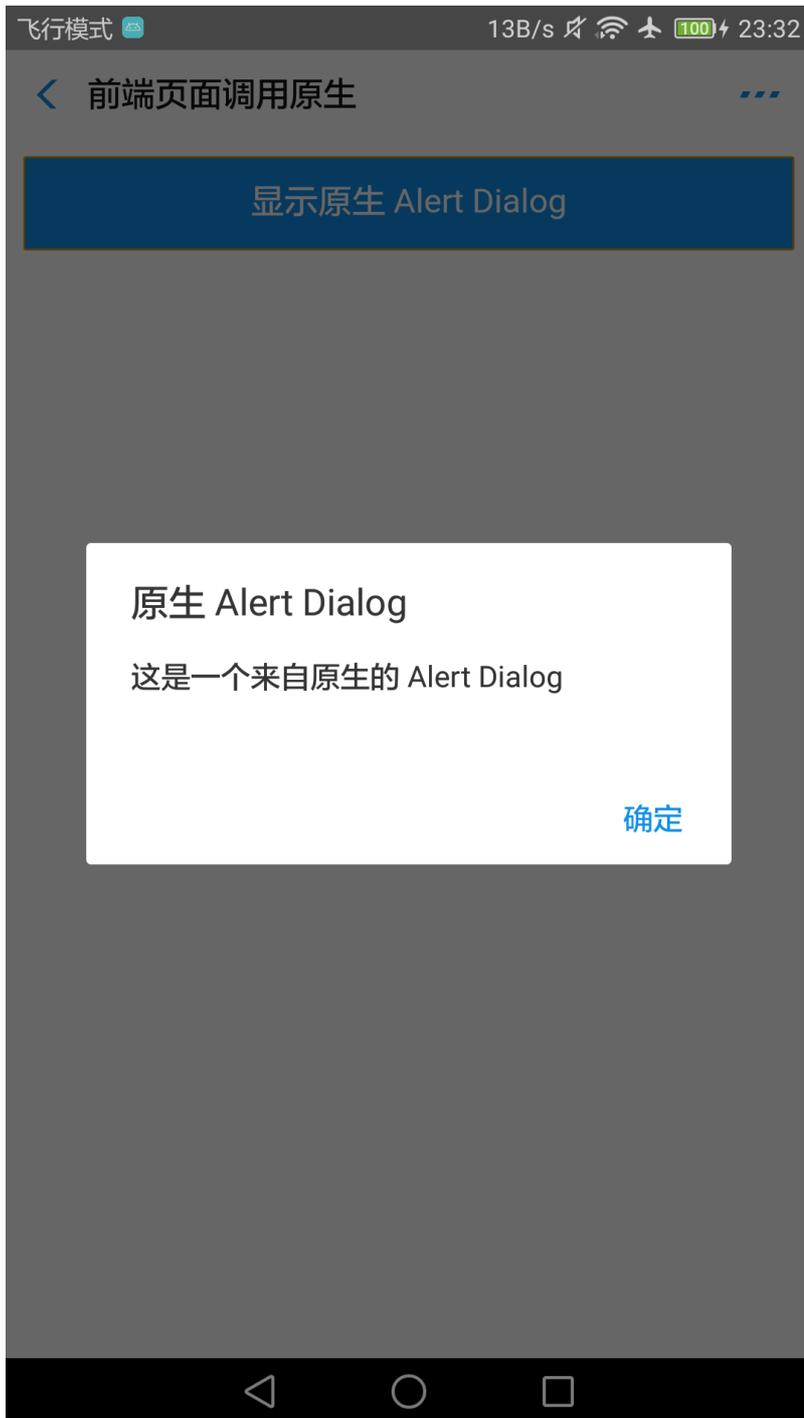
在 MainActivity 类定义点击按钮 h5_to_native_btn 后的行为，实现打开前端页面的功能。实现代码如下所示：

```
findViewById(R.id.h5_to_native_btn).setOnClickListener(new View.OnClickListener() {  
@Override  
public void onClick(View v) {  
MPNebula.startUrl("https://mcube-prod.oss-cn-hangzhou.aliyuncs.com/570DA89281533-default/80000000/1.0.0.1_all/nebula/fallback/h5_to_native.html");  
}  
});
```

编译工程后，在手机上安装应用。打开应用后界面如下：



点击按钮后即可打开前端页面，点击按钮 **显示原生 Alert Dialog**，会弹出原生的警示框，警示框的标题是 **原生 Alert Dialog**，消息框的内容是 **这是一个来自原生的 Alert Dialog**；点击警示框的 **确定** 按钮，会再弹出一个无标题警示框，内容是 **点击了按钮**。说明接口调用成功。至此，您已完成 **前端调用 Native 接口**。





前端调用自定义 JSAPI

点击 [查看视频](#)，查看对应的视频教程。

构建一个自定义类 MyJSApiPlugin，用来定义自定义的 JSAPI。

```
package com.mpaas.demo;
```

```

import com.alibaba.fastjson.JSONObject;
import com.alipay.mobile.h5container.api.H5BridgeContext;
import com.alipay.mobile.h5container.api.H5Event;
import com.alipay.mobile.h5container.api.H5EventFilter;
import com.alipay.mobile.h5container.api.H5SimplePlugin;

public class MyJSApiPlugin extends H5SimplePlugin {

    private static final String API = "myapi";

    @Override
    public void onPrepare(H5EventFilter filter) {
        super.onPrepare(filter);
        filter.addAction(API);
    }

    @Override
    public boolean handleEvent(H5Event event, H5BridgeContext context) {
        String action = event.getAction();
        if (API.equalsIgnoreCase(action)) {
            JSONObject params = event.getParam();
            String param1 = params.getString("param1");
            String param2 = params.getString("param2");
            JSONObject result = new JSONObject();
            result.put("success", true);
            result.put("message", API + " with " + param1 + ", " + param2 + " was handled by native.");
            context.sendBridgeResult(result);
            return true;
        }
        return false;
    }
}

```

在工程中注册自定义的 JSAPI——MyJSApiPlugin。推荐在应用启动的时候注册。此处我们注册在 MyApplication 中。

```

public class MyApplication extends Application {
    @Override
    protected void attachBaseContext(Context base) {
        super.attachBaseContext(base);
        // 建议判断下是否主进程，只在主进程初始化
        QuinoxlessFramework.setup(this, new IInitCallback() {
            @Override
            public void onPostInit() {
                // 在这里开始使用mPaaS功能
                //调用registerCustomJsapi()完成自定义 JSAPI的注册。
                registerCustomJsapi();
            }
        });
    }

    @Override
    public void onCreate() {
        super.onCreate();
    }
}

```

```

QuinoxlessFramework.init();
}

private void registerCustomJsapi(){
MPNebula.registerH5Plugin(
// 插件的 class name
MyJSApiPlugin.class.getName(),
// 填空即可
"",
// 作用范围，填"page"即可
"page",
// 注册的 jsapi 名称
new String[]{"myapi"});
}
}

```

在前端页面中，调用该自定义 JSAPI。示例如下：

```

AlipayJSBridge.call('myapi', {
param1: 'JsParam1',
param2: 'JsParam2'
}, function (result) {
alert(JSON.stringify(result));
});

```

说明： https://mcube-prod.oss-cn-hangzhou.aliyuncs.com/570DA89281533-default/80000001/1.0.0.1_all/nebula/fallback/custom_jsapi.html 是已经写好的前端页面，您可以调用此页面以体验前端调用自定义 JSAPI 接口的功能。

在 activity_main.xml 文件中，新增按钮 button，button id 为 custom_jsapi_btn。

```

<Button
android:id="@+id/custom_jsapi_btn"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="10dp"
android:background="#108EE9"
android:gravity="center"
android:text="自定义 JSAPI"
android:textColor="#ffffff"/>

```

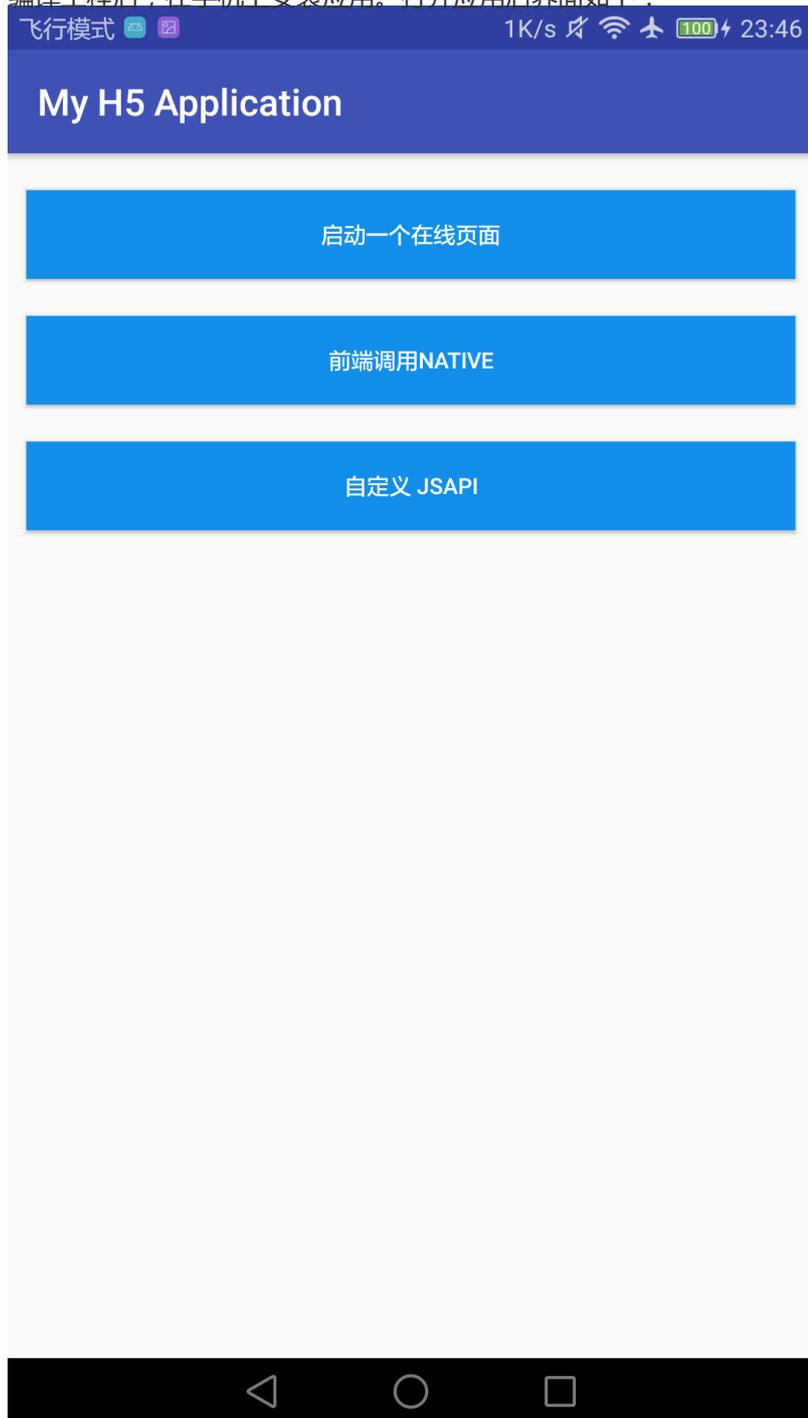
在 MainActivity 类定义点击按钮 custom_jsapi_btn 后的行为，实现前端调用自定义 JSAPI 接口的功能。实现代码如下所示：

```

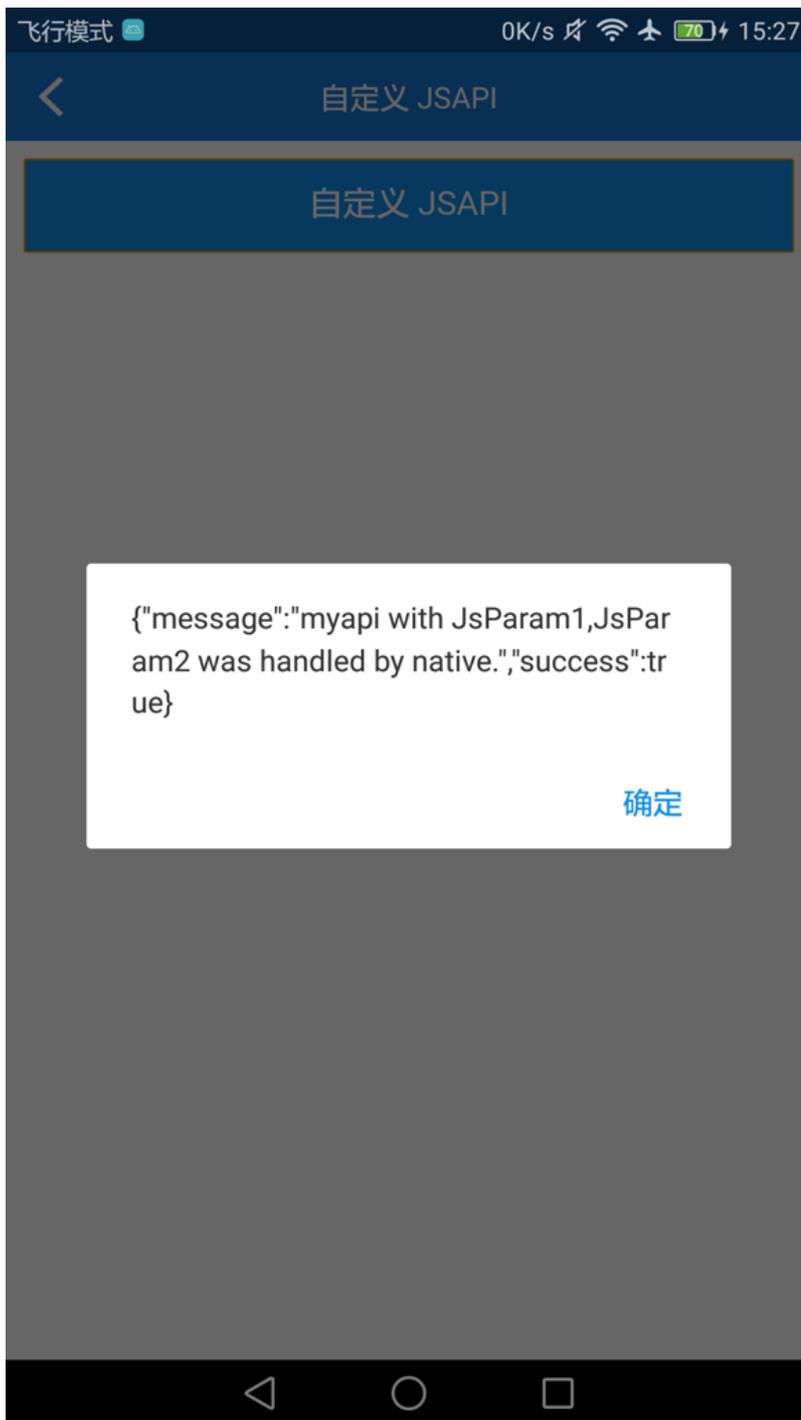
findViewById(R.id.custom_jsapi_btn).setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
MPNebula.startUrl("https://mcube-prod.oss-cn-hangzhou.aliyuncs.com/570DA89281533-default/80000001/1.0.0.1_all/nebula/fallback/custom_jsapi.html");
}
});

```

编译工程后，在手机上安装应用。打开应用后界面如下：



7. 点击按钮后即可打开前端页面，点击按钮 **自定义 JSAPI**，会打开包含了一个按钮 **自定义 JSAPI** 的前端页面。点击该 **自定义 JSAPI** 按钮，会再弹出一个无标题警示框，内容按照自定义 API 定义的功能处理了的前端调用时传入的参数。至此，您已完成 **前端调用 自定义 JSAPI 接口**。



自定义H5页面的 TitleBar

点击 [查看视频](#)，查看对应的视频教程。

H5 容器提供方法可以设置自定义的标题栏，您可以继承 mPaaS 提供的默认标题栏 `MpaasDefaultH5TitleView`，然后根据自己的需求，重写其中的一些方法。当然，您也可以自己实现 `H5TitleView`。在本教程中我们使用 `MpaasDefaultH5TitleView`。

构建一个 `H5ViewProvider` 实现类，在 `createTitleView` 方法中返回您定义的 `H5TitleView`。

```
package com.mpaas.demo.myh5application;

import android.content.Context;
import android.view.ViewGroup;

import com.alipay.mobile.nebula.provider.H5ViewProvider;
import com.alipay.mobile.nebula.view.H5NavMenuView;
import com.alipay.mobile.nebula.view.H5PullHeaderView;
import com.alipay.mobile.nebula.view.H5TitleView;
import com.alipay.mobile.nebula.view.H5WebContentView;
import com.mpaas.nebula.adapter.view.MpaasDefaultH5TitleView;

public class H5ViewProviderImpl implements H5ViewProvider {
    @Override
    public H5TitleView createTitleView(Context context) {
        return new MpaasDefaultH5TitleView(context);
    }

    @Override
    public H5NavMenuView createNavMenu() {
        return null;
    }

    @Override
    public H5PullHeaderView createPullHeaderView(Context context, ViewGroup viewGroup) {
        return null;
    }

    @Override
    public H5WebContentView createWebContentView(Context context) {
        return null;
    }
}
```

在 activity_main.xml 文件中，新增按钮 button，button id 为 custom_title_btn。

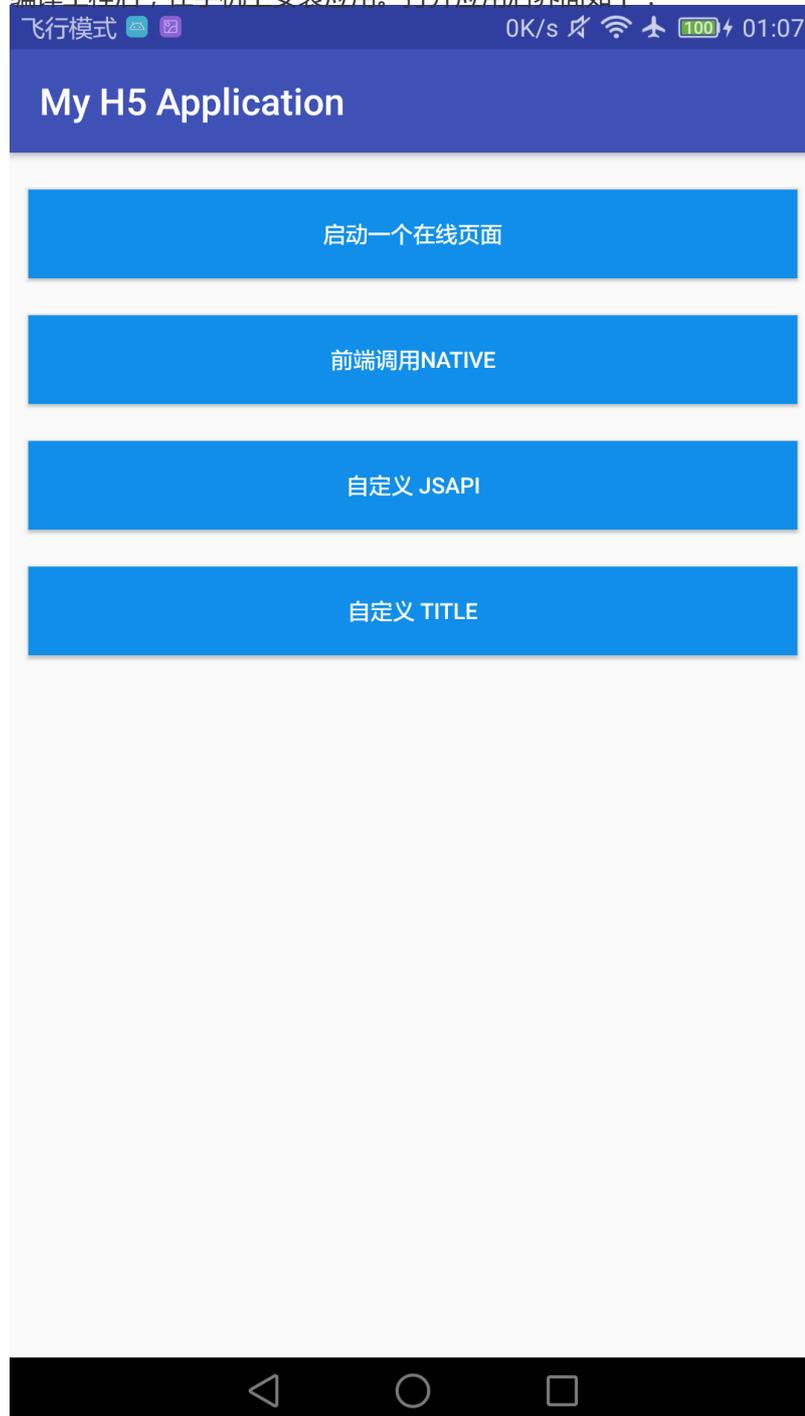
```
<Button
    android:id="@+id/custom_title_btn"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:background="#108EE9"
    android:gravity="center"
    android:text="自定义 Title"
    android:textColor="#ffffff"/>
```

在 MainActivity 类定义点击按钮 custom_title_btn 后的行为，将自定义 View Provider 设给容器，并打开一个在线网页。实现代码如下所示：

```
findViewById(R.id.custom_title_btn).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
```

```
// 设置自定义 title ( 设置一次即可 )
MPNebula.setCustomViewProvider(new H5ViewProviderImpl());
// 随意启动一个地址 , title 已经改变
MPNebula.startUrl("https://www.cloud.alipay.com/docs/2/49549");
}
});
```

编译工程后，在手机上安装应用。打开应用后界面如下：

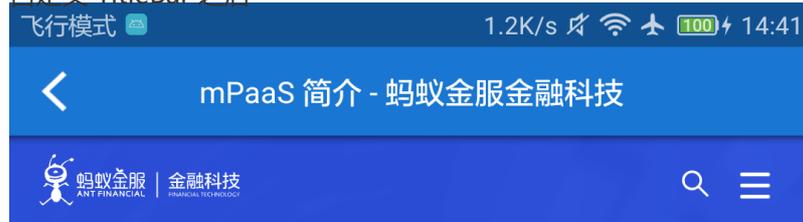


5. 点击按钮 **自定义TITLE**，**自定义 JSAPI**，会打开在线网页，可以看到此页面的 titlebar 的颜色、字体颜色都发生了变化。至此，您已完成 **自定义H5 页面的 TitleBar**。

- 自定义 TitleBar 之前



- 自定义 TitleBar 之后



文档 > 移动开发平台 > mPaaS 简介

全文档 ▾ Q 输入关键词

mPaaS 简介



更新时间: 2019-05-09 19:16

移动开发平台 (Mobile PaaS, 简称 mPaaS) 是源于支付宝 App 的移动开发平台, 为移动开发、测试、运营及运维提供云到端的一站式解决方案, 能有效降低技术门槛、减少研发成本、提升开发效率, 协助企业快速搭建稳定高质量的移动 App。



使用 UC 内核

在 Android 应用中接入 UC SDK 能够有效解决各种厂商浏览器的兼容性问题, 保持比系统浏览器更低闪退率并且性能卓越。使用 UC 内核前, 您需要申请 UCKey, 操作参考 使用 UC 内核。

2.7.7 使用 H5 离线包

H5 离线包的使用可以分为四个部分: 发布离线包、预置离线包、启动离线包和更新离线包。在本教程中为了说

明和演示 H5 离线包的功能，采取了从发布到预置，再到启动，最后到更新的流程，但是这一流程并非使用 H5 离线包的必要条件。在实际生产中，您可以自由地根据需要进行使用。

发布离线包

本节介绍了发布离线包的操作流程。

点击 [查看视频](#)，查看对应的视频教程。

前提条件

一个前端 App 的 zip 包（如果您没有自己的前端离线包，您可以下载我们为您准备好的 [示例离线包](#)）。

操作步骤

1. 您需要先在控制台的应用中配置离线包信息，请参考 [配置离线包](#)。
2. 您需要生成您自己的前端App的离线包（或者使用我们的示例离线包），请参考 [生成离线包](#)。
3. 您需要在控制台上创建该离线包并上传，请参考 [创建离线包](#)。
4. 您需要将配置好的离线包发布到您的客户端app中，请参考 [发布离线包](#)。

预置离线包

本节介绍了预置离线包的操作流程。

点击 [查看视频](#)，查看对应的视频教程。

前提条件

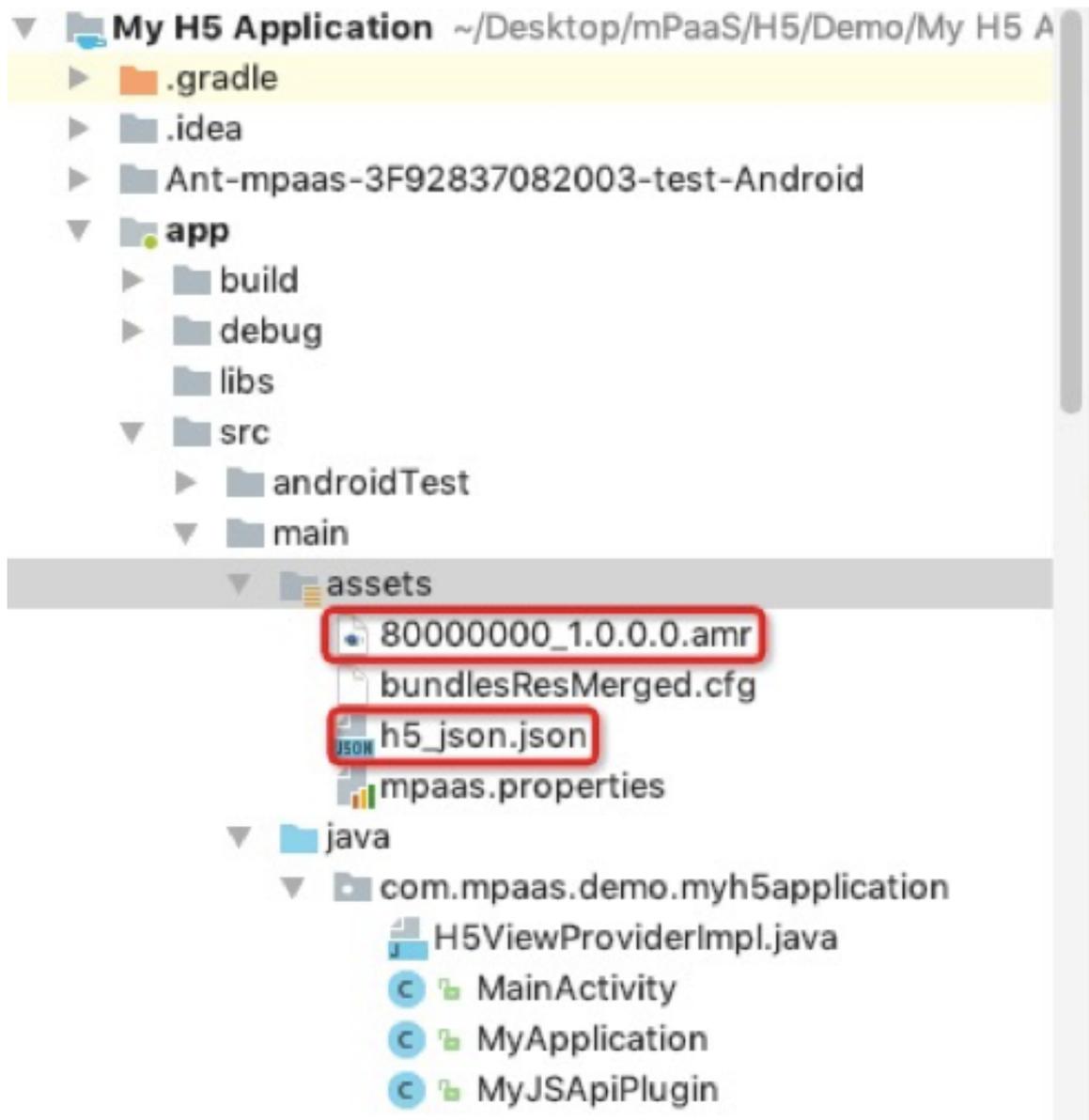
您已经在 mPaaS 控制台发布了离线包。

操作步骤

1. 在控制台下载离线包 AMR 文件和离线包配置文件到本地。



2. 将下载到的离线包 AMR 文件和离线包配置文件放在工程中的 assets 目录下。



将离线包预置到应用内。推荐在 app 启动的时候注册，在本教程中在 MyApplication 类中进行注册。至此，您已经完成 预置离线包。

```
public class MyApplication extends Application {
    @Override
    protected void attachBaseContext(Context base) {
        super.attachBaseContext(base);
        // 建议判断下是否主进程，只在主进程初始化
        QuinoxlessFramework.setup(this, new IInitCallback() {
            @Override
            public void onPostInit() {
                // 在这里开始使用mPaaS功能
                registerCustomJsapi();
                //调用loadOfflineNebula()以加载离线包。
                loadOfflineNebula();
            }
        });
    }
}
```

```
}

@Override
public void onCreate() {
    super.onCreate();
    QuinoxlessFramework.init();
}
private void loadOfflineNebula() {
    new Thread(new Runnable() {
        @Override
        public void run() {
            // 此方法为阻塞调用，请不要在主线程上调用内置离线包方法。如果内置多个amr包，要确保文件已存在，如不存在，会造成其他内置离线包失败。
            // 此方法仅能调用一次，多次调用仅第一次调用有效。
            MPNebula.loadOfflineNebula("h5_json.json", new
            MPNebulaOfflineInfo("80000000_1.0.0.0.amr","80000000","1.0.0.0"));
        }
    }).start();
}
}
```

启动离线包

本节介绍了更新离线包的操作流程。

点击 [查看视频](#)，查看对应的视频教程。

前提条件

您在客户端中已经预置了离线包。

操作步骤

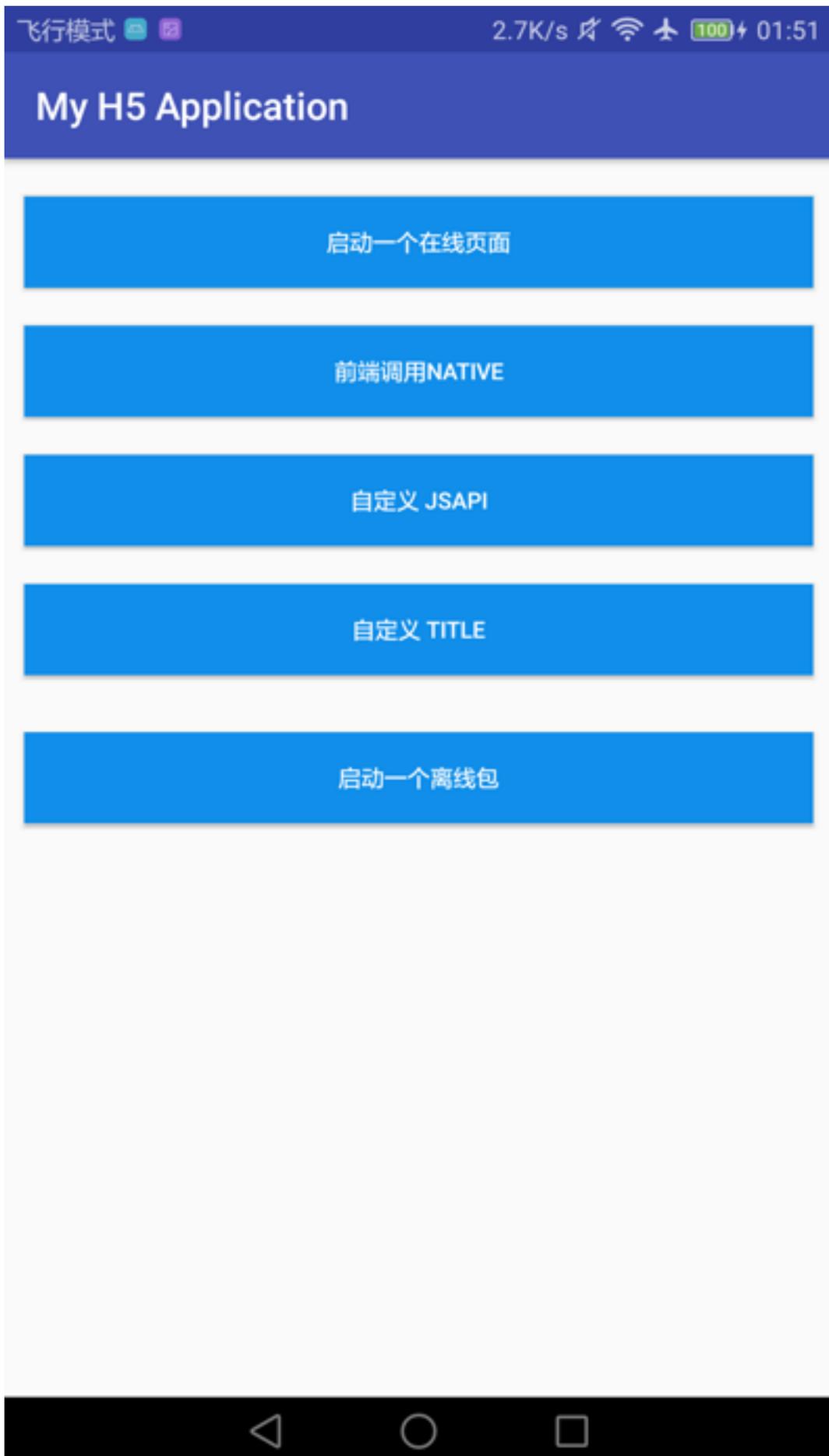
1. 在 activity_main.xml 文件中，新增按钮，button id 为 start_app_btn。

```
<Button
    android:id="@+id/start_app_btn"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="10dp"
    android:layout_marginLeft="10dp"
    android:layout_marginRight="10dp"
    android:layout_marginTop="20dp"
    android:background="#108EE9"
    android:gravity="center"
    android:text="启动一个离线包"
    android:textColor="#ffffff"/>
```

2. 在 MainActivity 类定义点击按钮start_app_btn后的行为，启动离线包。其中传入的参数“80000000”为离线包的 APPID。

```
findViewById(R.id.start_app_btn).setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        MPNebula.startApp("80000000");  
    }  
});
```

3. 编译工程后，在手机上安装应用。打开应用后界面如下。



4. 点击按钮 **启动一个离线包**，即可打开离线包中预置的网页。至此，您已完成 **启动离线包**。

飞行模式

5.5K/s 5G 100% 01:51

< 前端页面调用原生

显示原生 Alert Dialog



更新离线包

本节介绍了更新离线包的操作流程。

点击 [查看视频](#)，查看对应的视频教程。

前提条件

您的客户端应用中已经预置了离线包，并且在 mPaaS 控制台上已经创建了新的离线包版本，也上传了新版本的离线包。

操作步骤

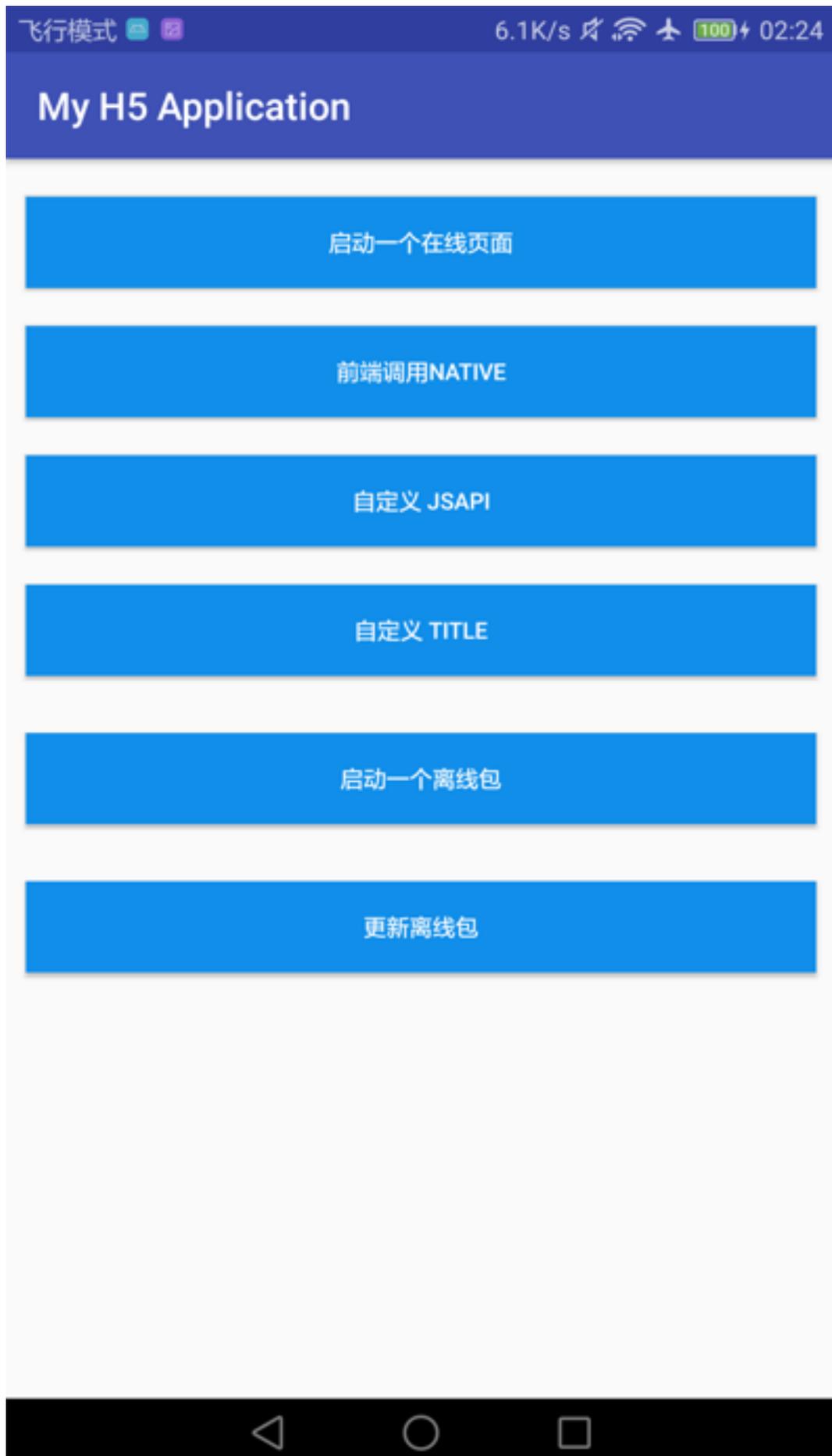
1. 在 activity_main.xml 文件中，新增按钮，button id 为 update_app_btn。

```
<Button
    android:id="@+id/update_app_btn"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="10dp"
    android:layout_marginLeft="10dp"
    android:layout_marginRight="10dp"
    android:layout_marginTop="20dp"
    android:background="#108EE9"
    android:gravity="center"
    android:text="更新离线包"
    android:textColor="#ffffff"/>
```

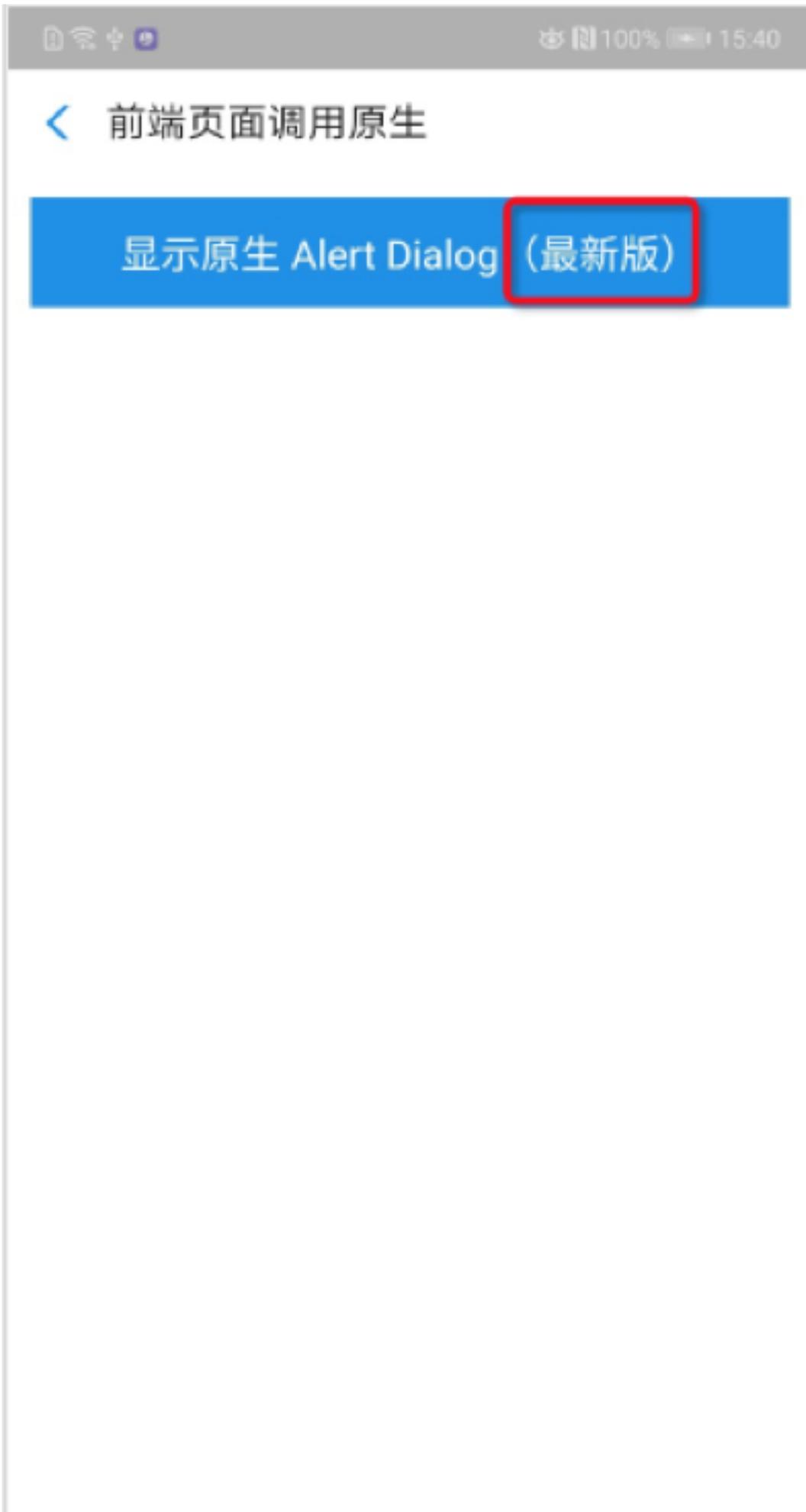
2. 在 MainActivity 类定义点击按钮 update_app_btn 后的行为，启动离线包。其中传入的参数“80000000”为离线包的 APPID。

```
findViewById(R.id.update_app_btn).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        MPNebula.updateAllApp(new MpaasNebulaUpdateCallback() {
            @Override
            public void onResult(final boolean success, final boolean isLimit) {
                // success 为是否成功
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        Toast.makeText(MainActivity.this, success ? "离线包更新成功":"离线包更新失败",
                            Toast.LENGTH_SHORT).show();
                    }
                });
            }
        });
    }
});
```

3. 编译工程后，在手机上安装应用。打开应用后界面如下。



4. 点击按钮 **更新离线包**，即可打开更新离线包。在提示更新成功后，在点击按钮 **启动一个离线包**，即可以看到更新后的页面。至此，您已完成 **更新离线包**。



2.8 Android 常见问题

自定义 JSAPI 时，`interceptEvent` 和 `handleEvent` 有什么区别？

解答：如果您要监听容器的其他事件，写在 `interceptEvent` 里；如果事件只是您自己处理的，写在 `handleEvent` 里，返回 `true` 就行。

如果返回 `true`，事件将停止传递，如果返回 `false`，事件将继续传递到其他插件。

自定义 JSAPI 时，已经在 `config.setEvents("event");` 添加了事件，为什么还要在插件的 `onPrepare` 里添加一次事件？

解答：因为容器的插件是懒加载的，即在页面创建的时候加载。通过外部的 `config.setEvents` 来注入要监听的事件名称，当真正有 JS 调用的时候，才会去实例化对应的插件对象。真正实例化的插件的事件分发用的是插件内 `onPrepare` 的事件。所以，要确保 `config.setEvents("event")` 的事件和内部 `onPrepare` 的事件保持一致。

自定义 JSAPI 插件注册的 `page`、`session`、`service` 三者有什么区别？

解答：`page` 对应一个 `webview`。`session` 对应 `mPaaS` 应用的一个 `App` 对象。`service` 是全局的一个单例。

- 如果注册为 `page` 级别，每次创建 `WebView` 都会创建一个插件实例，对应的插件的 `onRelease` 在 `WebView` 销毁时回调。
- 如果注册为 `session` 级别，每次创建一个 `App` 对象都会创建一个插件。
- 如果注册为 `service` 级别，全局只创建一次插件，在第一次打开容器的时候创建。

验签是验证离线包的来源吗？还是会做完整性校验？

解答：在离线包发布平台，如果您配置了签名私钥，平台下发的 `amr` 文件就会带上离线包的签名信息（通过私钥对离线包的 `hash` 值加密后得到的密文）。验签的时候会使用在项目中预置的公钥进行签名的验证（解密上一步的密文得到 `hash` 值，本地计算离线包的 `hash`，判断两者是否相同）。

该过程保证了即保证了离线包的内容完整，也保证了离线包的来源正确。

打开离线包 `ProgressBar` 无法隐藏？

解答：这种情况可能是由于资源离线失败，页面 `fallback` 到了线上模式，在 `fallback` 模式下面 H5 容器强制限制 `ProgressBar`，无法通过设置 `SHOW_PROGRESS` 启动参数来隐藏页面加载进度条（`Progress Bar`）。

3 接入 iOS

3.1 添加 iOS SDK

H5 容器支持 **基于 mPaaS 框架**、**基于原生框架且使用 mPaaS 插件** 2 种接入方式，您可以自由选择。根据选择的接入方式，您可以参考 **基于 mPaaS 框架接入** 或 **基于原生框架且使用 mPaaS 插件接入 H5 容器 SDK** 的添加。

3.2 使用容器

3.2.1 10.1.60 升级指南

初始化容器

- 初始化时机：需在框架加载之前调用，必须在 DTFrameworkInterface 的 - (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions 中调用。

```

37 - (BOOL)shouldAutoactivateShareKit
38 {
39     return YES;
40 }
41
42 - (DTNavigationBarBackTextStyle)navigationBarBackTextStyle
43 {
44     return DTNavigationBarBackTextStyleAlipay;
45 }
46
47 - (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
48 {
49     // 初始化容器
50     [MPNebulaAdapterInterface initNebula];
51
52     // NSString *presetApplistPath = [[NSBundle mainBundle] pathForResource:[NSString
53     stringWithFormat:@"DemoCustomPresetApps.bundle/NebulaApplist.plist"] ofType:nil];
54     // NSString *appPackagePath = [[NSBundle mainBundle] pathForResource:[NSString
55     stringWithFormat:@"DemoCustomPresetApps.bundle"] ofType:nil];
56     // NSString *pluginsJsapisPath = [[NSBundle mainBundle] pathForResource:[NSString
57     stringWithFormat:@"DemoCustomPlugins.bundle/Poseidon-UserDefine-Extra-Config.plist"] ofType:nil];
58     // [MPNebulaAdapterInterface initWithCustomPresetApplistPath:presetApplistPath
59     customPresetAppPackagePath:appPackagePath customPluginsJsapisPath:pluginsJsapisPath];
60 }
  
```

- 若已有工程基线为 10.1.32 :
 - 需修改自定义 JsApi 路径、预置离线包及包信息路径：

必须在 DTFrameworkInterface 的 - (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions 中调用 initNebulaWithCustomPresetApplistPath。同时，需要将 afterDidFinishLaunchingWithOptions 替换为 beforeDidFinishLaunchingWithOptions。

```

37 - (BOOL)shouldAutoactivateShareKit
38 {
39     return YES;
40 }
41
42 - (DTNavigationBarBackTextStyle)navigationBarBackTextStyle
43 {
44     return DTNavigationBarBackTextStyleAlipay;
45 }
46
47 - (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
48 {
49     // 初始化容器
50     [MPNebulaAdapterInterface initNebula];
51
52     NSString *presetApplistPath = [[NSBundle mainBundle] pathForResource:[NSString
53     stringWithFormat:@"DemoCustomPresetApps.bundle/NebulaApplist.plist"] ofType:nil];
54     NSString *appPackagePath = [[NSBundle mainBundle] pathForResource:[NSString
55     stringWithFormat:@"DemoCustomPresetApps.bundle"] ofType:nil];
56     NSString *pluginsJsapisPath = [[NSBundle mainBundle] pathForResource:[NSString
57     stringWithFormat:@"DemoCustomPlugins.bundle/Poseidon-UserDefine-Extra-Config.plist"] ofType:nil];
58     [MPNebulaAdapterInterface initWithCustomPresetApplistPath:presetApplistPath
59     customPresetAppPackagePath:appPackagePath customPluginsJsapisPath:pluginsJsapisPath];
60 }
  
```

- 需指定所有 H5 页面的基类/全局资源包/UA/是否验签等配置：

需在容器初始化之后调用，必须在 DTFrameworkInterface 的 - (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions 中设置，否则会被容器默认配置覆盖。

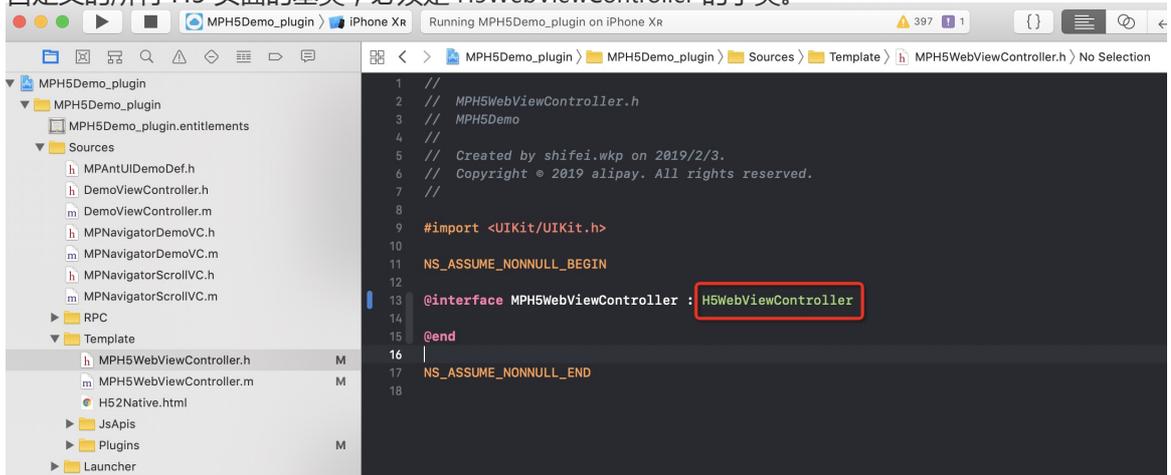
```

< > Portal > MPaaS > Targets > Portal > APMo...ework > m DTFrameworkInterface+Portal.m M -application:afterDidFinishLaunchingWithOptions: < >
78
79 // 启动netctrl
80 // [[MASSAccess sharedInstance] dispatchTokenSvrInfo:nil uploadInfo:nil downloadInfo:nil downloadInfoCrypto:nil
configStorage:nil];
81
82 // 小程序初始化
83 // [MPTinyAppAdapterInterface initWithTinyApp];
84
85 // Nebula容器初始化
86 NSString *customPresetApplistPath = [[NSBundle mainBundle].bundlePath stringByAppendingFormat:@"%/%@/%@",
@"MPH5Demo.bundle", @"h5_json.json"];
87 NSString *customPresetAppPackagePath = [[NSBundle mainBundle].bundlePath stringByAppendingFormat:@"%/%@",
@"MPH5Demo.bundle"];
88 NSString *customPluginsJsapisPath = [[NSBundle mainBundle].bundlePath stringByAppendingFormat:@"%/%@/%@",
@"MPH5Demo.bundle", @"Poseidon-Custom-Config.plist"];
89 [MPNebulaAdapterInterface initWithNebulaWithCustomPresetApplistPath:customPresetApplistPath
customPresetAppPackagePath:customPresetAppPackagePath customPluginsJsapisPath:customPluginsJsapisPath];
90 }
91
92 = (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions ⚠ Categ...
93 {
94 [MPNebulaAdapterInterface sharedInstance].nebulaVeiwControllerClass = NSClassFromString(@"MPH5WebViewController");
95 [MPNebulaAdapterInterface sharedInstance].nebulaNeedVerify = NO;
96 [MPNebulaAdapterInterface sharedInstance].nebulaUserAgent = @"mPaaS/Portal";
97 [MPNebulaAdapterInterface sharedInstance].nebulaCommonResourceApplist = @[@"7777777"];
98
99 // 全量更新
100 [[MPNebulaAdapterInterface sharedInstance] requestAllNebulaApps:^(NSDictionary *data, NSError *error) {
101 |
102 |
103 |
104 |
}
}

```

容器基类

- 自定义的所有 H5 页面的基类，必须是 H5WebViewController 的子类。



```

1 //
2 // MPH5WebViewController.h
3 // MPH5Demo
4 //
5 // Created by shifei.wkp on 2019/2/3.
6 // Copyright © 2019 alipay. All rights reserved.
7 //
8
9 #import <UIKit/UIKit.h>
10
11 NS_ASSUME_NONNULL_BEGIN
12
13 @interface MPH5WebViewController : H5WebViewController
14
15 @end
16
17 NS_ASSUME_NONNULL_END
18

```

- 若原有 H5 基类中实现了 back 方法，需删除。

```

107 }
108 }
109
110 - (void)viewDidAppear:(BOOL)animated
111 {
112     [super viewDidAppear:animated];
113 }
114
115 - (void)viewWillDisappear:(BOOL)animated
116 {
117     [super viewWillDisappear:animated];
118 }
119
120 // #pragma mark - back
121 // - (void)back
122 // {
123 //     // 触发back事件
124 //     id<PSDPluginProtocol> navigationItemPlugin = [[self psdScene] pluginManager]
125 //         plugin:@"NBPlugin4NavigationItem"];
126 //     if ([navigationItemPlugin respondsToSelector:@selector(backItemClicked:)] {
127 //         [navigationItemPlugin performSelector:@selector(backItemClicked:) withObject:nil];
128 //     }
129 // }

```

自定义导航栏

- 返回按钮：需监听 `kNBEvent_Scene_NavigationItem_Left_Back_Create_Before` 事件，修改默认导航栏样式

```

63 [self.target addEventListener:kEvent_Proxy_Request_Start_Handler withListener:self useCapture:NO];
64
65 [super pluginDidLoad];
66 }
67
68 - (void)handleEvent:(PSDEvent *)event
69 {
70     [super handleEvent:event];
71
72     if ([kNBEvent_Scene_NavigationItem_Left_Back_Create_Before isEqualToString:event.eventType]) {
73         // 在默认返回按钮基础上，修改样式
74         NSArray *leftBarButtonItems = event.context.currentViewController.navigationItem.leftBarButtonItems;
75         if ([leftBarButtonItems count] == 1) {
76             if ([leftBarButtonItems[0] isKindOfClass:[AUIButton class]]) {
77                 // 在默认返回按钮基础上，修改返回箭头和文案颜色
78                 AUIButton *backItem = leftBarButtonItems[0];
79                 backItem.backButtonColor = [UIColor greenColor];
80                 backItem.titleColor = [UIColor colorWithHexString:@"#00ff00"];
81             }
82         }
83     }
84     [event preventDefault];
85     [event stopPropagation];

```

3.2.2 使用 SDK (版本 ≥ 10.1.60)

本文将结合 [H5容器和离线包](#) 官方 Demo 来介绍10.1.60 以上版本的基线中 H5 容器 SDK 的使用。

初始化容器

启动容器

为了使用 Nebula 容器，您需要在程序启动完成后调用 SDK 接口，对容器进行初始化。必须在 `DTFrameworkInterface` 的 `-(void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions` 中进行初始化。

```

- (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{

```

```
// 初始化容器
[MPNebulaAdapterInterface initNebula];
}
```

若您需要使用 **预置离线包**、**自定义 JsApi** 和 **Plugin** 等功能，请将上方代码中的 `initNebula` 替换为下方代码中的 `initNebulaWith` 接口，传入对应参数对容器进行初始化。

- `presetApplistPath`：自定义的预置离线包的包信息路径。
- `appPackagePath`：自定义的预置离线包的包路径。
- `pluginsJsapisPath`：自定义 JsApi 和 Plugin 文件的存储路径。

```
- (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
// 初始化容器
NSString *presetApplistPath = [[NSBundle mainBundle] pathForResource:[NSString stringWithFormat:@"MPCustomPresetApps.bundle/h5_json.json"] ofType:nil];
NSString *appPackagePath = [[NSBundle mainBundle] pathForResource:[NSString stringWithFormat:@"MPCustomPresetApps.bundle"] ofType:nil];
NSString *pluginsJsapisPath = [[NSBundle mainBundle] pathForResource:[NSString stringWithFormat:@"Poseidon-UserDefine-Extra-Config.plist"] ofType:nil];
[MPNebulaAdapterInterface initNebulaWithCustomPresetApplistPath:presetApplistPath
customPresetAppPackagePath:appPackagePath customPluginsJsapisPath:pluginsJsapisPath];
}
```

说明：`initNebula` 和 `initNebulaWithCustomPresetApplistPath` 是两个并列的方法，不要同时调用。

定制容器

如有需要，您可以通过设置 `MPNebulaAdapterInterface` 的属性值来定制容器配置。必须在 `DTFrameworkInterface` 的 `-(void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions` 中设置，否则会被容器默认配置覆盖。

```
- (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
// 定制容器
[MPNebulaAdapterInterface sharedInstance].nebulaVeiwControllerClass = [MPH5WebViewController class];
[MPNebulaAdapterInterface sharedInstance].nebulaNeedVerify = NO;
[MPNebulaAdapterInterface sharedInstance].nebulaUserAgent = @"mPaaS/Portal";
[MPNebulaAdapterInterface sharedInstance].nebulaCommonResourceAppList = @[@"7777777"];
}
```

属性含义如下：

名称	含义	备注
<code>nebulaVeiwControllerClass</code>	H5 页面的基类	默认为 <code>H5WebViewController</code> 。若需指定所有 H5 页面的基类，可直接设置此接口。注意：基类必须继承自 <code>H5WebViewController</code> 。

nebulaWebViewClass	设置 WebView 的基类	>10.1.60 : 默认为 H5WKWebView。自定义的 WebView 必须继承 H5WKWebView。 = 10.1.60 : 不支持自定义。
nebulaUseWKArbitrary	设置是否使用 WKWebView 加载离线包页面	>10.1.60 : 默认为 YES。 = 10.1.60 : 默认为 NO。
nebulaUserAgent	设置应用的 UserAgent	设置的 UserAgent 会作为后缀添加到容器默认的 UA 上。
nebulaNeedVerify	是否验签, 默认为 YES	若配置离线包时未上传私钥文件, 此值需设为 NO, 否则离线包加载失败。
nebulaPublicKeyPath	离线包验签的公钥	与配置离线包时上传的私钥对应的公钥。
nebulaCommonResourceAppList	公共资源包的 appId 列表	-
errorHtmlPath	当 H5 页面加载失败时展示的 HTML 错误页路径	默认读取 MPNebulaAdapter.bundle/error.html。
configDelegate	设置自定义开关 delegate	提供全局修改容器默认开关值的能力。

更新离线包

启动完成后, 全量请求所有离线包信息, 检查服务端是否有更新包。为了不影响应用启动速度, 建议在 (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions 之后调用。

```

- (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
// 定制容器
[MPNebulaAdapterInterface sharedInstance].nebulaVeivControllerClass = [MPH5WebViewController class];
[MPNebulaAdapterInterface sharedInstance].nebulaNeedVerify = NO;
[MPNebulaAdapterInterface sharedInstance].nebulaUserAgent = @"mPaaS/Portal";
[MPNebulaAdapterInterface sharedInstance].nebulaCommonResourceAppList = @[@"7777777"];

// 全量更新离线包
[[MPNebulaAdapterInterface sharedInstance] requestAllNebulaApps:^(NSDictionary *data, NSError *error) {
NSLog(@"");
}];
}

```

初始化完成后, 效果如下:

```

6
7 - (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
8 {
9     // 初始化 rpc
10    [MPRpcInterface initRpc];
11
12    // 初始化容器
13    // [MPNebulaAdapterInterface initNebula];
14
15    // 自定义jsapi路径和预置离线包信息
16    NSString *presetAppListPath = [[NSBundle mainBundle] pathForResource:[NSString
17    stringWithFormat:@"DemoCustomPresetApps.bundle/NebulaAppList.plist"] ofType:nil];
18    NSString *appPackagePath = [[NSBundle mainBundle] pathForResource:[NSString stringWithFormat:@"DemoCustomPresetApps.bundle"]
19    ofType:nil];
20    NSString *pluginsJsapisPath = [[NSBundle mainBundle] pathForResource:[NSString
21    stringWithFormat:@"DemoCustomPlugins.bundle/Poseidon-UserDefine-Extra-Config.plist"] ofType:nil];
22    [MPNebulaAdapterInterface initNebulaWithCustomPresetAppListPath:presetAppListPath customPresetAppPackagePath:appPackagePath
23    customPluginsJsapisPath:pluginsJsapisPath];
24 }
25
26 - (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
27 {
28     // 定制容器
29    [MPNebulaAdapterInterface sharedInstance].nebulaVeiwControllerClass = [MPH5WebViewController class]; //设置H5容器基类
30    [MPNebulaAdapterInterface sharedInstance].nebulaUserAgent = @"mPaaS/Portal"; //设置H5容器UserAgent
31    [MPNebulaAdapterInterface sharedInstance].nebulaUseWKArbitrary = YES; //开启 WKWebView
32    [MPNebulaAdapterInterface sharedInstance].nebulaCommonResourceAppList = @[@"7777777"]; // 设置全局资源包
33    [MPNebulaAdapterInterface sharedInstance].nebulaNeedVerify = NO; // 关闭离线包验签, 正式版本请开启验签
34
35    // 更新离线包
36    [[MPNebulaAdapterInterface sharedInstance] requestAllNebulaApps:^(NSDictionary *data, NSError *error) {
37        NSLog(@"mpaas nebula rpc data :%@", data);
38    }];
39 }
40
41 @end

```

非框架托管配置

如下图所示, 若您 App 的生命周期并没有交给 mPaaS 框架托管, 而是指定为您自己定义的 delegate, 那么还需进行额外配置。若是 mPaaS 框架托管, 可调过此步骤。

```

< > MPH5Demo_pod > MPH5Demo_pod > main.m > No Selection
1 //
2 // main.m
3 // MPH5Demo_pod
4 //
5 // Created by yangwei on 2019/3/26.
6 // Copyright © 2019 yangwei. All rights reserved.
7 //
8
9 #import <UIKit/UIKit.h>
10 #import "AppDelegate.h"
11
12 int main(int argc, char * argv[]) {
13     [MPAnalysisHelper enableCrashReporterService]; // USE MPAAS CRASH REPORTER
14     @autoreleasepool {
15         return UIApplicationMain(argc, argv, nil, NSStringFromClass([AppDelegate class]));
16     }
17     return UIApplicationMain(argc, argv, @"DFApplication", @"DFClientDelegate"); // NOW USE MPAAS FRAMEWORK
18 }
19

```

启动 mPaaS 框架

在当前应用的 didFinishLaunchingWithOptions 方法中调用 `[[DTFrameworkInterface sharedInstance] manualInitMpaasFrameworkWithApplication:application launchOptions:launchOptions];` 来启动 mPaaS 框架。

```

14 @end
15
16 @implementation AppDelegate
17
18
19 + (AppDelegate *)sharedInstance
20 {
21     return [UIApplication sharedApplication].delegate;
22 }
23
24 - (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
25
26     self.window = [[UIWindow alloc] initWithFrame:[UIScreen mainScreen].bounds];
27     self.window.rootViewController = [[DFNavigationController alloc]
28         initWithRootViewController:[[TestDemoViewController alloc] init]];
29     self.navigationController = self.window.rootViewController;
30     [self.window makeKeyAndVisible];
31     self.window.backgroundColor = [UIColor whiteColor];
32
33     // navigationController创建完成后, 启动 mPaaS 框架
34     [[DTFrameworkInterface sharedInstance] manualInitMpaasFrameworkWithApplication:application
35         launchOptions:launchOptions];
36
37     return YES;
38 }

```

说明：启动框架必须在当前应用 window 和 navigation 初始化完成后调用，否则无法生效。

创建应用启动器

创建继承 DTBootLoader 的子类，重写 createWindow 和 createNavigationController 方法，返回当前应用自己的 window 和 navigationController。

- 设置 window：当前应用的 keyWindow。
- 设置 navigationController：当前应用 keyWindow 的 rootviewController，必须继承 DFNavigationController。

```

1 //
2 // MPBootLoaderImpl.h
3 // Portal
4 //
5 // Created by yemingyu on 2019/6/24.
6 // Copyright © 2019 Alibaba. All rights reserved.
7 //
8
9 #import <APMobileFramework/APMobileFramework.h>
10
11 NS_ASSUME_NONNULL_BEGIN
12
13 @interface MPBootLoaderImpl : DTBootLoader
14
15 @end
16
17 NS_ASSUME_NONNULL_END
18

```

```

< > MPH5Demo_pod > MPaaS > Targets > MPH5Demo_pod > APMobileFramework > m MPBootLoaderImpl.m > No...ction
1 //
2 // MPBootLoaderImpl.m
3 // Portal
4 //
5 // Created by yemingyu on 2019/6/24.
6 // Copyright © 2019 Alibaba. All rights reserved.
7 //|
8
9 #import "MPBootLoaderImpl.h"
10 #import "AppDelegate.h"
11
12 @implementation MPBootLoaderImpl
13
14 - (UINavigationController *)createNavigationController
15 {
16     return [AppDelegate sharedInstance].navigationController;
17 }
18
19 - (UIWindow *)createWindow
20 {
21     return [AppDelegate sharedInstance].window;
22 }
23
24 @end

```

指定应用启动器

在 DTFrameworkInterface 的 category 中重写方法，指定当前应用自己的 bootloader，并隐藏 mPaaS 框架默认的 window 和 launcher 应用。

```

< > MPH5Demo_pod > MPaaS > Ta...ets > M...od > AP...rk > m DTFrameworkInterface+MPH5Demo_pod.m > No Selection
5     }];
6 }
7
8 #pragma mark 非框架托管配置
9 - (DTBootLoader *)bootLoader {
10     static MPBootLoaderImpl *_bootLoader;
11     static dispatch_once_t onceToken;
12     dispatch_once(&onceToken, ^{
13         _bootLoader = [[MPBootLoaderImpl alloc] init];
14     });
15     return _bootLoader;
16 }
17
18 - (BOOL)shouldWindowMakeVisable {
19     return NO;
20 }
21
22 - (BOOL)shouldShowLauncher {
23     return NO;
24 }
25
26 @end

```

唤起容器

容器初始化完成后，就可以唤起一个 H5 容器。分为以下三种情况：

基于在线 URL 或本地 HTML 文件，创建一个 H5 容器。示例代码如下：

```
// 打开在线 URL
[[MPNebulaAdapterInterface sharedInstance] startH5ViewControllerWithParams:@{@"url":
@"https://tech.antfin.com/products/MPAAS"}];
// 打开本地 HTML 页面
NSString *path = [[NSBundle mainBundle].bundlePath stringByAppendingFormat:@"%/%@/%@",
@"MPH5Demo.bundle", @"H52Native.html"];
if ([path length] > 0) {
[[MPNebulaAdapterInterface sharedInstance] startH5ViewControllerWithParams:@{@"url": path}];
}
```

基于传入的离线包信息，创建一个 H5 容器，并自动 push 打开。示例代码如下：

```
[[MPNebulaAdapterInterface sharedInstance]
startH5ViewControllerWithNebulaApp:@{@"appId":@"90000000"}];
```

基于传入的离线包信息，创建一个 H5 容器，并返回创建的 H5 容器实例（一般用在首页 tab 页面）。示例代码如下：

```
[[MPNebulaAdapterInterface sharedInstance] createH5ViewControllerWithNebulaApp:@{@"appId":@"90000000"}];
```

实现 H5 与 Native 双向通信

您可以通过调用 JSAPI 和监听特定事件来实现 H5 与 Native 双向通信。

在 H5 页面调用 Native 功能

您可以通过调用 JSAPI 来实现 H5 到 Native 的通信。

Nebula 容器支持的 JSAPI 及相关参数说明，请参见 内置 JSAPI 。

示例

例如，您可以调用 JSAPI 接口 `pushWindow`，以实现“在 H5 页面点击某个按钮时，加载一个新页面”的需求：

```
AlipayJSBridge.call('pushWindow', {
url: 'https://tech.antfin.com',
param: {
readTitle: true,
defaultTitle: true,
// ...
}
}, function(data) {alert('调用结果'+JSON.stringify(data)); });
```

AlipayJSBridge 说明

AlipayJSBridge 是 Nebula 容器自动注入的 JSBridge。在 `Window.onload` 以后，容器会生成一个全局变量 `AlipayJSBridge`，然后触发 `AlipayJSBridgeReady` 事件。AlipayJSBridge 注入是一个异步过程，因此需要先监听 `AlipayJSBridgeReady` 事件再调用接口。

示例代码如下：

```
<h1>bridge 使用方法</h1>

<script>
function ready(callback) {
if (window.AlipayJSBridge) {
callback && callback();
} else {
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}

ready(function(){
alert('bridge ready');
});
</script>
```

在 Native 页面调用 H5 功能

您可以通过监听特定事件来实现 Native 到 H5 的通信。Nebula 容器支持的事件列表，参见 [事件扩展](#)。

```
document.addEventListener('back', function (e) {
if(confirm('back已拦截,是否确定返回?')) {
// do something;
}
}, false);
```

除了 Nebula 容器默认支持的事件外，您还可以在 Native 端通过以下方式自定义事件让前端来监听。

```
// self: 当前 H5 页面所在的 VC
// data : native 传递给前端的参数
// callBack : 前端收到事件后的回调
[self callHandler:@"customEvent" data:@{@"key":@"value"} responseCallback:^(id responseData) {
NSLog(@"前端收到事件后的回调: %@", responseData);
}];
```

扩展 Nebula 容器能力

如果 Nebula 容器提供的基础 H5 页面双向通信能力无法满足需求，您可以对 Nebula 进行扩展开发：

JSAPI：如果您要从 H5 页面发起 Native 功能调用（如显示一个 ActionSheet，或显示联系人对话框），那么您需要扩展 JSAPI。JSAPI 可以通过 handler 方法，让您很方便地增加 H5 页面的 Native 功能调用来实现特定功能。具体的自定义方法，请参见 [自定义 JSAPI](#)。

Plugin：如果您需要在某个时机（如进入页面、收到请求等）做某些事情（如记录埋点、修改返回数据等），那么您需要开发一个插件 (Plugin)。在插件中订阅相应的事件后，就可以在 handler 中对事件所携带的数据进行加工处理。具体的自定义方法，请参见 [自定义插件](#)。

加载离线包

传统的在线 H5 技术容易受到网络环境影响，从而影响 H5 页面的性能。为了最大程度摆脱网络对 H5 页面加载的影响，您可以将不同的业务封装打包成为一个离线包，通过发布平台下发到客户端，对客户端资源进行更新。更多信息，请参见 [离线包简介](#) 和 [使用离线包](#)。

H5 容器埋点

在 H5 页面加载时，Nebula 容器会自动监控加载性能，并捕获相关的行为数据和异常报错数据。更多信息，请参见 [H5 容器埋点](#)。

3.2.3 使用 SDK (版本 = 10.1.32)

本文将向您介绍介绍 10.1.32 基线下 H5 容器 SDK 的使用。

初始化容器

启动容器

为了使用 Nebula 容器，您需要在程序启动完成后调用 SDK 接口，对容器进行初始化。必须在 DTFrameworkInterface 的 - (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions 中进行初始化。

```
- (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
// 初始化容器
[MPNebulaAdapterInterface initNebula];
}
```

若您需要使用 [预置离线包](#)、[自定义 JsApi](#) 和 [Plugin](#) 等功能，只能存储在以下默认 bundle 中，否则不生效。

名称	含义
MPCustomPlugins.bundle	自定义的 JSAPI 和 Plugin 路径
MPCustomPresetApps.bundle	预置的离线包和包信息路径

定制容器

如有需要，您可以通过设置 MPNebulaAdapterInterface 的属性值来定制容器配置。必须在 DTFrameworkInterface 的 - (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions 中设置，否则会被容器默认配置覆盖。

```
- (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
// 初始化容器
[MPNebulaAdapterInterface initNebula];
}
```

```
// 定制容器
[MPNebulaAdapterInterface sharedInstance].nebulaVeivControllerClass = [MPH5WebViewController class];
[MPNebulaAdapterInterface sharedInstance].nebulaNeedVerify = NO;
[MPNebulaAdapterInterface sharedInstance].nebulaUserAgent = @"mPaaS/Portal";
[MPNebulaAdapterInterface sharedInstance].nebulaCommonResourceAppList = @[@"7777777"];
}
```

属性含义如下：

名称	含义	备注
nebulaVeivControllerClass	H5 页面的基类	默认为 UIViewController。若需指定所有 H5 页面的基类，可直接设置此接口。
nebulaUserAgent	设置应用的 UserAgent	设置的 UserAgent 会作为后缀添加到容器默认的UA上
nebulaNeedVerify	是否验签，默认为 YES	若 配置离线包 时未上传私钥文件，此值需设为 NO，否则离线包加载失败。
nebulaPublicKeyPath	离线包验签的公钥	与 配置离线包 时上传的私钥对应的公钥。
nebulaCommonResourceAppList	公共资源包的 appId 列表	-
errorHtmlPath	当 H5 页面加载失败时展示的 HTML 错误页路径	默认读取 MPNebulaAdapter.bundle/error.html。

更新离线包

启动完成后，全量请求所有离线包信息，检查服务端是否有更新包。为了不影响应用启动速度，建议在 (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions 之后调用。

```
- (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
// 初始化容器
[MPNebulaAdapterInterface initNebula];

// 定制容器
[MPNebulaAdapterInterface sharedInstance].nebulaVeivControllerClass = [MPH5WebViewController class];
[MPNebulaAdapterInterface sharedInstance].nebulaNeedVerify = NO;
[MPNebulaAdapterInterface sharedInstance].nebulaUserAgent = @"mPaaS/Portal";
[MPNebulaAdapterInterface sharedInstance].nebulaCommonResourceAppList = @[@"7777777"];

// 全量更新离线包
[[MPNebulaAdapterInterface sharedInstance] requestAllNebulaApps:^(NSDictionary *data, NSError *error) {
NSLog(@"");
}];
}
```

初始化完成后，效果如下：

```

< > MPH5Demo_pod > ... > ... > ... > ... > DTFrameworkInterface+MPH5Demo_pod.m > M -application:afterDidFinishLaunchingWithOptions: < > >
}
- (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // 初始化容器
    [MPNebulaAdapterInterface sharedInstance] initNebula;

    // 定制容器
    [MPNebulaAdapterInterface sharedInstance].nebulaVeivControllerClass = [MPH5WebViewController class]; //设置H5容器基类
    [MPNebulaAdapterInterface sharedInstance].nebulaUserAgent = @"mPaaS/Portal"; //设置H5容器UserAgent
    [MPNebulaAdapterInterface sharedInstance].nebulaUseWKArbitrary = YES; //开启 WKWebView
    [MPNebulaAdapterInterface sharedInstance].nebulaCommonResourceAppList = @[@"7777777"]; // 设置全局资源包
    [MPNebulaAdapterInterface sharedInstance].nebulaNeedVerify = NO; // 关闭离线包验签, 正式版本请开启验签

    // 更新离线包
    [[MPNebulaAdapterInterface sharedInstance] requestAllNebulaApps:^(NSDictionary *data, NSError *error) {
        NSLog(@"[mPaaS] nebula rpc data :%@", data);
    }];
}
@end

```

唤起容器

容器初始化完成后, 就可以唤起一个 H5 容器。分为以下三种情况:

基于在线 URL 或本地 HTML 文件, 创建一个 H5 容器。示例代码如下:

```

// 打开在线 URL
[[MPNebulaAdapterInterface sharedInstance] startH5ViewControllerWithParams:@{@"url":
@"https://tech.antfin.com/products/MPAAS"}];
// 打开本地 HTML 页面
NSString *path = [[NSBundle mainBundle].bundlePath stringByAppendingFormat:@"%s/%s/%s",
@"MPH5Demo.bundle", @"H52Native.html"];
if ([path length] > 0) {
    [[MPNebulaAdapterInterface sharedInstance] startH5ViewControllerWithParams:@{@"url": path}];
}

```

基于传入的离线包信息, 创建一个 H5 容器, 并自动 push 打开。示例代码如下:

```

[[MPNebulaAdapterInterface sharedInstance]
startH5ViewControllerWithNebulaApp:@{@"appId":@"90000000"}];

```

基于传入的离线包信息, 创建一个 H5 容器, 并返回创建的 H5 容器实例 (一般用在首页 tab 页面) 。示例代码如下:

```

[[MPNebulaAdapterInterface sharedInstance] createH5ViewControllerWithNebulaApp:@{@"appId":@"90000000"}];

```

实现 H5 与 Native 双向通信

您可以通过调用 JSAPI 和监听特定事件来实现 H5 与 Native 双向通信。

在 H5 页面调用 Native 功能

您可以通过调用 JSAPI 来实现 H5 到 Native 的通信。

Nebula 容器支持的 JSAPI 及相关参数说明，请参见 内置 JSAPI 。

示例

例如，您可以调用 JSAPI 接口 pushWindow，以实现“在 H5 页面点击某个按钮时，加载一个新页面”的需求：

```
AlipayJSBridge.call('pushWindow', {
  url: 'https://tech.antfin.com',
  param: {
    readTitle: true,
    defaultTitle: true,
    // ...
  }
}, function(data) {alert('调用结果'+JSON.stringify(data)); });
```

AlipayJSBridge 说明

AlipayJSBridge 是 Nebula 容器自动注入的 JSBridge。在 Window.onload 以后，容器会生成一个全局变量 AlipayJSBridge，然后触发 AlipayJSBridgeReady 事件。AlipayJSBridge 注入是一个异步过程，因此需要先监听 AlipayJSBridgeReady 事件再调用接口。

示例代码如下：

```
<h1>bridge 使用方法</h1>

<script>
function ready(callback) {
  if (window.AlipayJSBridge) {
    callback && callback();
  } else {
    document.addEventListener('AlipayJSBridgeReady', callback, false);
  }
}

ready(function(){
  alert('bridge ready');
});
</script>
```

在 Native 页面调用 H5 功能

您可以通过监听特定事件来实现 Native 到 H5 的通信。Nebula 容器支持的事件列表，参见 事件扩展 。

```
document.addEventListener('back', function (e) {
  if(confirm('back已拦截,是否确定返回?')) {
    // do something;
  }
}, false);
```

除了 Nebula 容器默认支持的事件外，您还可以在 Native 端通过以下方式自定义事件让前端来监听

```
// self: 当前 H5 页面所在的 VC
// data : native 传递给前端的参数
// callBack : 前端收到事件后的回调
[self callHandler:@"customEvent" data:@{@"key":@"value"} responseCallback:^(id responseData) {
    NSLog(@"前端收到事件后的回调: %@", responseData);
}];
```

扩展 Nebula 容器能力

如果 Nebula 容器提供的基础 H5 页面双向通信能力无法满足需求，您可以对 Nebula 进行扩展开发：

JSAPI：如果您要从 H5 页面发起 Native 功能调用（如显示一个 ActionSheet，或显示联系人对话框），那么您需要扩展 JSAPI。JSAPI 可以通过 handler 方法，让您很方便地增加 H5 页面的 Native 功能调用来实现特定功能。具体的自定义方法，请参见 自定义 JSAPI。

Plugin：如果您需要在某个时机（如进入页面、收到请求等）做某些事情（如记录埋点、修改返回数据等），那么您需要开发一个插件 (Plugin)。在插件中订阅相应的事件后，就可以在 handler 中对事件所携带的数据进行加工处理。具体的自定义方法，请参见 自定义插件。

加载离线包

传统的在线 H5 技术容易受到网络环境影响，从而影响 H5 页面的性能。为了最大程度摆脱网络对 H5 页面加载的影响，您可以将不同的业务封装打包成为一个离线包，通过发布平台下发到客户端，对客户端资源进行更新。更多信息，请参见 离线包简介 和 使用离线包。

H5 容器埋点

在 H5 页面加载时，Nebula 容器会自动监控加载性能，并捕获相关的行为数据和异常报错数据。更多信息，请参见 H5 容器埋点。

3.3 使用离线包

3.3.1 管理离线包

传统的在线 H5 技术容易受到网络环境影响，从而影响 H5 页面的性能。为此，您可以将不同的业务封装打包成一个离线包，通过发布平台下发对客户端资源进行更新。

本文将引导您管理离线包：

- 生成离线包
- 加载离线包
- 动态更新离线包

前置条件

您需要确保完成 SDK 添加后，客户端工程已集成 NebulamPaaSBiz.framework。

生成离线包

为了生成 .amr 离线包, 您需要构建前端 .zip 包并在线生成 .amr 包。具体的操作步骤, 请参见文档 生成离线包。

加载离线包

根据是否在客户端预置离线包, 分为以下两种方式：

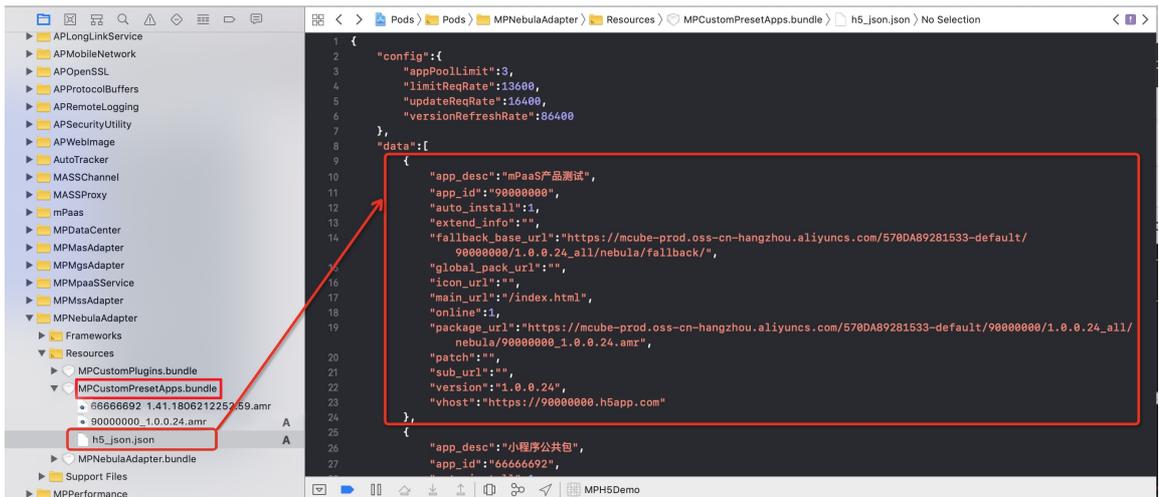
- 预置离线包
- 加载远程离线包

预置离线包

不管网络情况如何, 首页或登录页等页面都需快速加载。因此, 可以将这部分资源打包为离线包、提前预置在工程中, 以保证在离线情况下资源也能快速加载。

具体步骤如下：

将从发布平台下载的 .amr 离线包和 h5_json.json 文件添加到工程 MPCustomPresetApps.bundle 目录下：



重要：目前发布平台仅支持下载单个离线包的 h5_json.json 配置文件；当预置多个离线包时，需要手动合并 JSON 文件中 data 数组。

进入对应的页面时，调用 Nebula 容器提供的接口方法加载离线包。

代码示例

```

- (void)openPresetPackage {
[[MPNebulaAdapterInterface sharedInstance]
startH5ViewControllerWithNebulaApp:@{@"appId":@"20180910"}];
}

```

接口方法

```

@interface MPNebulaAdapterInterface : NSObject

/**
 * 基于传入的离线包信息，创建一个 H5 容器，并自动 push 打开
 *
 * @param params H5容器的启动参数，必填参数：appId，其他可选参数参考文档 https://tech.antfin.com/docs/2/85001
 */
- (void)startH5ViewControllerWithNebulaApp:(NSDictionary *)params;

@end

```

加载远程离线包

除将离线包预置在客户端外，您还可以通过发布平台动态发布一个离线包。此时，客户端直接加载远程离线包，可以避免因预置大量离线包而使得客户端包体积过大。

具体步骤如下：

在应用启动完成后，可以预加载获取包信息并下载离线包，防止打开离线包时白屏。

代码示例

```

[[MPNebulaAdapterInterface sharedInstance] requestAllNebulaApps:^(NSDictionary *data,
NSError *error) {
NSLog(@"[mpaas] nebula rpc data :%@", data);
}];

```

接口方法

```

@interface MPNebulaAdapterInterface : NSObject

/**
 * 全量更新本地离线包信息
 *
 * @param finish 完成回调
 */
- (void)requestAllNebulaApps:(NAMRequestFinish)finish;

/**
 * 单个应用请求
 *
 * @param params 请求列表，格式为{appid:version}。可传多个 appid。版本号最长 4 位，如 1.0.0.1；不指定 version 时传空，默认取最高版本；支持版本号模糊匹配，如： '*' 匹配最高版本号、 '1.*' 匹配以 1 开头的版本号中最高的版本号
 * @param finish 完成回调
 */
- (void)requestNebulaAppsWithParams:(NSDictionary *)params finish:(NAMRequestFinish)finish;

```

```
@end
```

客户端配置完成后，您可以通过发布平台下发一个离线包，具体的操作步骤，请参见 [实时发布 > 离线包管理 > 发布离线包](#)。

进入对应的页面时，调用 Nebula 容器提供的接口方法加载离线包，即可看到您在发布平台下发的离线包。

代码示例

```
- (void)openPresetPackage {
[[MPNebulaAdapterInterface sharedInstance]
startH5ViewControllerWithNebulaApp:@{@"appId":@"20180910"}];
}
```

接口方法

```
@interface MPNebulaAdapterInterface : NSObject

/**
 * 基于传入的离线包信息，创建一个 H5 容器，并自动 push 打开
 *
 * @param params H5 容器的启动参数，必填参数：appId，其他可选参数参考文档
 https://tech.antfin.com/docs/2/85001
 */
- (void)startH5ViewControllerWithNebulaApp:(NSDictionary *)params;

@end
```

动态更新离线包

mPaaS 提供强大的动态更新能力，您可以通过发布平台下发一个新版本的离线包，来更新客户端的对应页面。具体的操作步骤，请参见 [实时发布 > 离线包管理 > 发布离线包](#)。

相关链接

- [离线包介绍](#)
- [代码示例](#)

3.4 自定义 JSAPI

要从页面发起 Native 功能调用，例如显示一个 ActionSheet，或显示联系人对话框，您需要扩展一个 JavaScript API (JSAPI)。使用 JSAPI，可以让您在 H5 页面增加 Native 功能调用入口。通过实现自定义 JSAPI 类中的 handler 方法，以 Native 的形式实现特定功能。

H5 容器组件提供以下能力：

- 丰富的内置 JSAPI，实现例如页面 push、pop、标题设置等功能。更多信息，查看 [参考：内置 JSAPI](#)。

- 支持用户自定义 JSAPI 和插件功能，扩展业务需求。

本文介绍如何实现自定义 JSAPI 功能。

关于此任务

自定义一个 JSAPI 可以有以下两种方式：

- Plist 注册
- 代码注册

操作步骤

Plist 注册

创建 JSAPI 类：

- 命名规范：为与容器默认提供的插件命名保持一致，创建的 JSAPI 类命名以 XXJsApi4 开头，其中 XX为自定义的前缀。
- 基类：所有 JSAPI 均继承自 PSDJsApiHandler。

实现基础方法：在 .m 文件中，需重写方法 `-(void)handler:context:callback:`。当前端调用此 JSAPI 时，会转发到此方法。

```
< > NebulaDemo > NebulaDemo > 自定义扩展插件 > H5JsApi4DemoTest.h > @interface H5JsApi4DemoTest
//
// H5JsApi4DemoTest.h
// NebulaDemo
//
// Created by Glance on 16/12/20.
// Copyright © 2016年 Alipay. All rights reserved.
//
@interface H5JsApi4DemoTest : PSDJsApiHandler
@end
```

```

< > NebulaDemo > NebulaDemo > 自定义扩展插件 > H5JsApi4DemoTest.m > No Selection
//
// H5JsApi4DemoTest.m
// NebulaDemo
//
// Created by Glance on 16/12/20.
// Copyright © 2016年 Alipay. All rights reserved.
//

#import "H5JsApi4DemoTest.h"

@implementation H5JsApi4DemoTest

- (void)handler:(NSDictionary *)data
  context:(PSDContext *)context
  callback:(PSDJsApiResponseCallbackBlock)callback
{
  [super handler:data context:context callback:callback];
  UIViewController *vc = context.currentViewController;
  UIWebView *webView = (UIWebView *)context.currentViewControllerProxy.psdContentView;
  NSLog(@"[NEBULADEMO]:JSAPI调用, 当前viewController %@, webView:%@, jsApi实例为 %@",vc, webView, self);

  if (!data) {
    errorCallback(callback, e_invalid_params);
    return;
  }
  else {
    callback(@"{"result":"jsapi-demoTest来自Native的处理结果"}");
  }
}

@end

```

该方法的参数含义如下：

参数名	含义
data	H5 页面调用此 JSAPI 时传入的参数
context	当前 H5 页面的上下文，具体可参考 PSDContext.h
callback	调用此 JSAPI 完成后的回调方法，以字典方式传递调用结果到 H5 页面

注册 JSAPI。在自定义的 Plist 文件中注册此 JSAPI。更多具体内容，查看 接入 iOS。

为统一管理自定义的 JSAPI 和 Plugin，新建一个 plsit 文件，内容如下。在 JsApis 数组下注册上一步创建的 JSAPI 类：

NebulaDemo > Resources > DemoPlugins.bundle > Poseidon-Extra-Config.plist

Key	Type	Value
▼ Root	Dictionary	(2 items)
▼ JsApiRuntime	Dictionary	(1 item)
▼ JsApis	Array	(2 items)
▼ Item 0	Dictionary	(2 items)
jsApi	String	demoTest
name	String	H5JsApi4DemoTest
▶ Item 1	Dictionary	(2 items)
▼ PluginRuntime	Dictionary	(1 item)
▶ Plugins	Array	(2 items)

- 注册的 JSAPI 是一个字典类型，包含以下两项内容：

名称	含义
jsApi	在 H5 页面中调用的 JSAPI 接口名
name	创建的 JSAPI 类名

同时需在初始化容器配置时，指定自定义的 Plist 文件的路径。

- 在 $\geq 10.1.32$ 版本中初始化 H5 容器，参见 [管理 H5 容器 > 版本 \$\geq 10.1.32\$](#) 。

```
// 自定义插件配置
NSString *plist_filePath = [[NSBundle mainBundle].bundlePath stringByAppendingFormat:@"%/%@/%@",
@"DemoPlugins.bundle", @"Poseidon-Extra-Config.plist"];
[NBServiceConfigurationGet() setExtraPluginsFilePath:plist_filePath];
```

代码注册

除使用 Plist 方式自定义一个 JSAPI 外，容器也支持直接调用 Nebula 容器提供的接口方法注册一个自定义的 JSAPI。更多具体内容，查看 [代码示例 H5Plugin4DemoTest](#) 类里的 `addJSApis` 方法：

```
``objective-c
// 这种 JSAPI 注册是在 plist 中配置之外的另一种方式
PSDJsApi *jsApi4DemoTest2 = [PSDJsApi jsApi:@"demoTest2"
handler:^(NSDictionary *data, PSDContext *context, PSDJsApiResponseCallbackBlock responseCallbackBlock) {
responseCallbackBlock(@{@ "result": @"jsapi-demoTest2调用 Native 的处理结果"});
}
checkParams:NO
isPrivate:NO
scope:self.scope];
[self registerJsApi2Target:jsApi4DemoTest2];
``
```

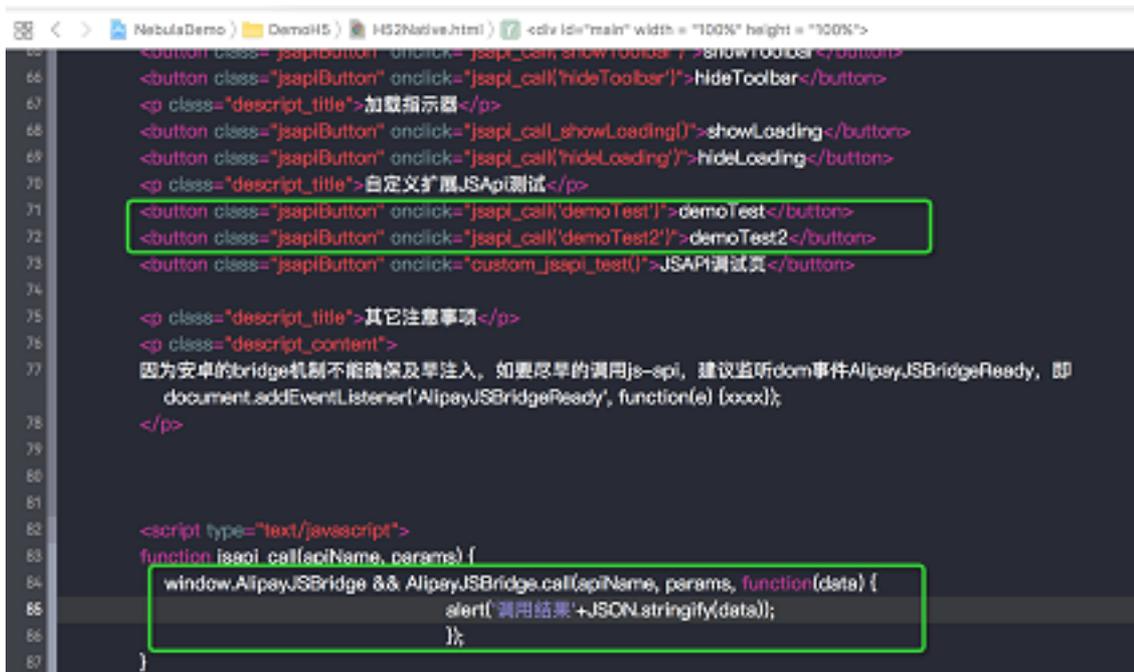
- 下表列出注册时各参数及其含义：

参数	含义
----	----

jsApi	在 H5 页面中调用的 JSAPI 接口名
handler	JSAPI 处理函数，功能同 Plist 注册方式中的 handler 方法
checkParams	是否检查参数
isPrivate	是否私有 JSAPI
scope	作用域

后续操作

在 H5 页面中调用自定义的 JSAPI：

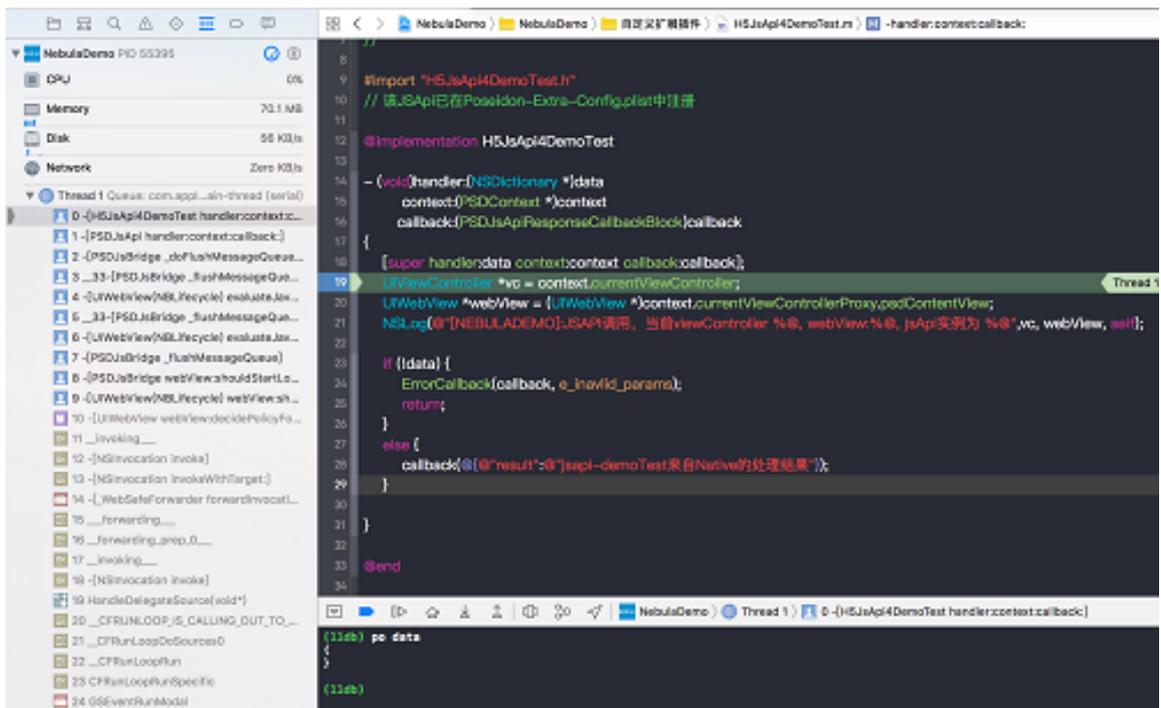


```

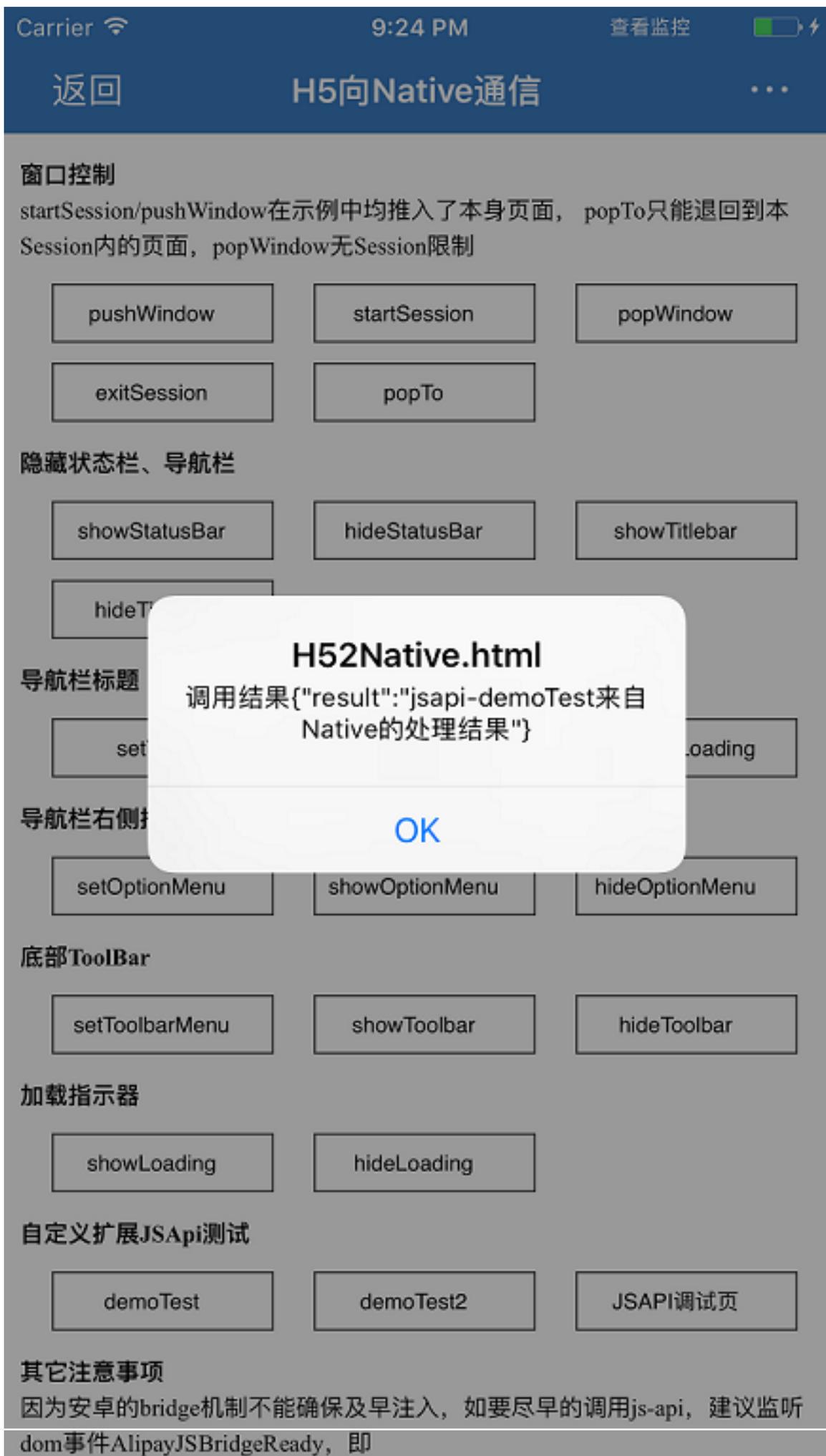
65 <button class="jsapiButton" onclick="jsapi_call('showToolBar')">showToolBar</button>
66 <button class="jsapiButton" onclick="jsapi_call('hideToolBar')">hideToolBar</button>
67 <p class="descript_title">加载指示器</p>
68 <button class="jsapiButton" onclick="jsapi_call('showLoading')">showLoading</button>
69 <button class="jsapiButton" onclick="jsapi_call('hideLoading')">hideLoading</button>
70 <p class="descript_title">自定义扩展JSApi测试</p>
71 <button class="jsapiButton" onclick="jsapi_call('demoTest')">demoTest</button>
72 <button class="jsapiButton" onclick="jsapi_call('demoTest2')">demoTest2</button>
73 <button class="jsapiButton" onclick="custom_jsapi_test()">JSAPI测试页</button>
74
75 <p class="descript_title">其它注意事项</p>
76 <p class="descript_content">
77 因为安卓的bridge机制不能确保及早注入，如果尽早的调用js-api，建议监听dom事件AlipayJSBridgeReady，即
78 document.addEventListener('AlipayJSBridgeReady', function(e) {xxx});
79
80
81
82 <script type="text/javascript">
83 function jsapi_call(apiName, params) {
84     window.AlipayJSBridge && AlipayJSBridge.call(apiName, params, function(data) {
85         alert('调用结果'+JSON.stringify(data));
86     });
87 }

```

在 handler 方法中添加断点，观察 H5 页面提供的参数是否符合预期：



3. 查看 H5 页面返回的结果是否符合预期：



相关链接

- 内置 JSAPI : 启动参数
- 代码示例

3.5 自定义插件

如果您需要在某个时机完成某些事件，例如进入页面时记录埋点，那么您需要开发一个插件 (Plugin)。插件订阅相应的事件后，就可以在 handler 中实现对事件所携带的数据进行加工处理。

关于此任务

自定义插件的过程分为以下步骤：

1. 新建 Plugin 。
2. 注册 Plugin 。
3. 使用 Plugin 。

查看 代码示例，自定义一个在 H5 页面加载时，修改页面导航栏的插件。

操作步骤

新建 Plugin

新创建的 Plugin 类一般格式如下，需注意以下几点：

- 命名：为与容器默认提供的 Plugin 命名保持一致，统一以 XXPlugin4 开头，其中 XX为自定义的前缀；
- 基类：所有插件均继承自 NBPluginBase；
- 实现基础方法：在 .m文件中，需重写以下三个方法。
 - - (void)pluginDidLoad：必选。监听的 H5 事件，事件列表请查看头文件 NBDefines.h。
 - - (void)addJSApis：可选。因为要与 H5 通信，可能需要注册 JSAPI。
 - - (void)handleEvent：必选。处理监听的事件触发后的逻辑。

.h 文件如下：

SHMetro > SHMetro > LSKit > H5 > H5Plugin4TitleView.h > No Selection

```
1 //
2 // H5Plugin4TitleView.h
3 // NebulaDemo
4 //
5 // Created by Glance on 16/12/16.
6 // Copyright © 2016年 Alipay. All rights reserved.
7 //
8
9 #import <NebulaSDK/NBPluginBase.h>
10
11 @interface H5Plugin4TitleView : NBPluginBase
12
13 @end
14
```

.m 文件如下：

< > SHMetro > SHMetro > LSKit > H5 > H5Plugin4TitleView.m > No Selec

```
//
// H5Plugin4TitleView.m
// NebulaDemo
//
// Created by Glance on 16/12/16.
// Copyright © 2016年 Alipay. All rights reserved.
//

#import "H5Plugin4TitleView.h"

@implementation H5Plugin4TitleView

- (void)pluginDidLoad
{ ... }

- (void)addJSApis
{ ... }

- (void)handleEvent:(NBNavigationTitleViewEvent *)event
{ ... }
```

监听事件

在 - (void)pluginDidLoad 方法中注册需要监听的事件。

```
- (void)pluginDidLoad {
self.scope = kPSDScope_Scene; // 1
[self.target
addEventListener:kNBEvent_Scene_TitleView_Title_Click // 2
withListener:self // 3
useCapture:NO]; // 4
[super pluginDidLoad];
}
```

addEventListener 方法用于监听某个事件，各参数说明如下：

名称	含义
scope	设置事件影响范围。目前支持的范围从小到大依次是 Scene、Session 和 Service。
event	设置事件名称，事件常量定义在 NBDefines.h 中。
listener	设置事件的处理者，即提供 - handleEvent: 的对象。

capture	设置是否使用捕捉的方式传播事件，一般使用 NO。
---------	--------------------------

添加 JSAPI

若在注册 Plugin 的过程中，需要自定义 JSAPI 与 H5 页面进行交互，可在 - (void)addJSApis 方法中，使用代码注册（参考 自定义 JSAPI > 代码注册）的方式进行处理。此方法为可选项，若无需要可不实现。

```
- (void)addJSApis
{
    [super addJSApis];
    // 可以在这里添加 TitleView 相关的自定义 JSAPI
}
```

处理监听

最后，在 - handleEvent: 中处理监听的事件触发后的逻辑。

```
- (void)handleEvent:(NBNavigationTitleViewEvent *)event
{
    [super handleEvent:event];

    if([kNBEvent_Scene_TitleView_Create_Before isEqualToString:event.eventType]) {
        // 自定义创建 TitleView，将失去默认 kNBEvent_Scene_TitleView_[Title_Set | Title_Click | Subtitle_Click] 等事件支持
        NBNavigationTitleViewEvent *e = (NBNavigationTitleViewEvent *)event;
        H5WebViewController *currentViewController = (H5WebViewController *)event.context.currentViewController;
        UINavigationController *navi = currentViewController.navigationController ?: (UINavigationController
        *)APPDELEGATE.window.rootViewController;
        UINavigationController *bar = navi.navigationController;
        H5NavigationTitleView *newTitleView = [self createNavigationTitleView:bar.bounds];
        [newTitleView setMainTitle:@"重新创建"subtitle:nil];
        newTitleView.delegate = [e.titleView delegate];
        currentViewController.navigationItem.titleView = newTitleView;
        e.titleView = newTitleView;
        [e preventDefault];
    }
}
```

说明：由于所有在 - (void)pluginDidLoad 方法中监听过的事件，触发时的处理都会集中到 - handleEvent: 方法中，所以如果监听了多个方法，需根据 event 的 class 或 eventType 对不同事件做不同处理。

注册 Plugin

创建了 Plugin 类后，需要在自定义的 Plist 文件（参见 自定义 JSAPI > 注册 JSAPI 说明）中注册此 Plugin。

Key	Type	Value
▼ Root	Dictionary	(2 items)
▶ JsApiRuntime	Dictionary	(1 item)
▼ PluginRuntime	Dictionary	(1 item)
▼ Plugins	Array	(2 items)
▼ Item 0	Dictionary	(3 items)
name	String	H5Plugin4DemoTest
scope	String	scene
▼ events	Array	(1 item)
▼ Item 0	Dictionary	(2 items)
name	String	-
useCapture	Boolean	NO
▶ Item 1	Dictionary	(3 items)

注册的 Plugin 是一个字典类型，包含以下 3 项内容：

名称	含义
name	创建的 Plugin 类名
scope	Plugin 生效的范围
events	监听的事件名称

使用 Plugin

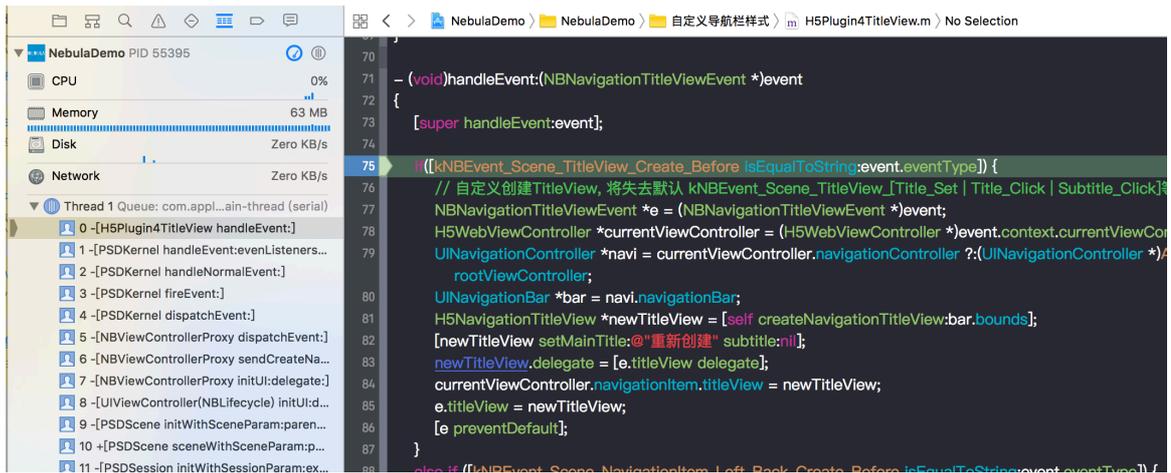
在 pluginDidLoad 方法中添加断点，观察触发时机的堆栈调用顺序是否正确。

```

1 //
2 // H5Plugin4TitleView.m
3 // NebulaDemo
4 //
5 // Created by Glance on 16/12/16.
6 // Copyright © 2016年 Alipay. All rights reserved.
7 //
8
9 #import "H5Plugin4TitleView.h"
10 #import "H5NavigationTitleView.h"
11 #import "H5WebViewController.h"
12
13 @implementation H5Plugin4TitleView
14
15 - (void)pluginDidLoad
16 {
17     self.scope = kPSDScope_Scene;
18     if ([DemoConfigServer sharedInstance].useCustomNavigationItemView) {
19         return;
20     }
21     // -- 如需重写事件支持，可监听如下事件重新创建
22     [self.target addEventListener:kNBEvent_Scene_TitleView_Create_Before
23         withListener:self
24         useCapture:NO];
25 }

```

在 handleEvent 方法中添加断点，观察监听的事件能否正确触发。



观察 H5 页面自定义的导航栏样式是否生效。



3.6 使用教程

3.6.1 总览

H5 容器是一款移动端 Hybrid 解决方案 SDK (Nebula SDK)，提供了良好的外部扩展功能，拥有功能插件化、事件机制、JSAPI 定制和 H5App 推送更新管理能力。离线包是将包括 HTML、Javascript、CSS 等页面内静态资源打包到一个压缩包内，通过预先下载该压缩包到本地，然后通过客户端打开，直接从本地加载离线包，从而最大程度地摆脱网络环境对 H5 页面的影响。本教程将带您一起体验和使用 H5 容器和离线包的基本能

力。

本教程一共分为3个部分：在 Xcode 创建工程，使用 H5 容器和使用 H5 离线包。

您将学会

- 如何创建一个 iOS 应用。
- 如何使用 mPaaS H5 容器服务。
- 如何使用 mPaaS H5 离线包服务。

您将需要

1. 配置开发环境。参考 [准备配置](#) 获取更多信息。
2. 网络访问。
3. 网络浏览器（建议您使用 Chrome 浏览器）。

3.6.2 在 mPaaS 控制台创建应用并下载配置文件

1. 打开网络浏览器，登录 [mPaaS 控制台](#)。
2. 创建 mPaaS 应用。

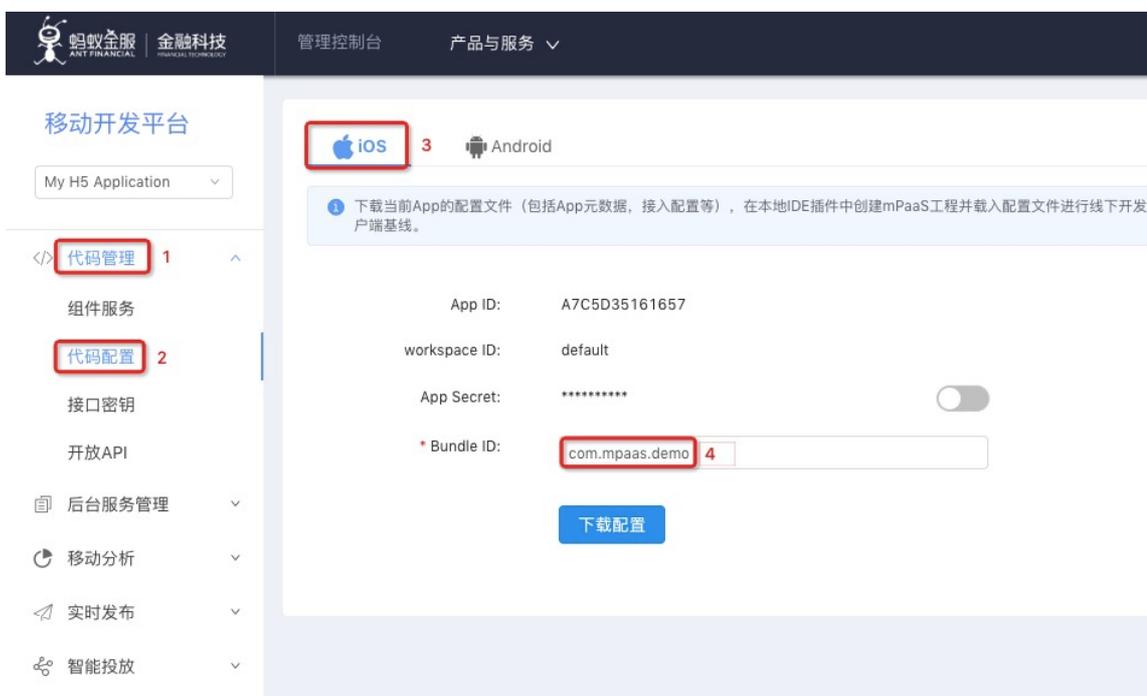


3. 输入项目名称并点击 **确定**。



点击刚创建的应用，进入应用。

点击 **代码管理** > **代码配置** > **iOS**，输入 **Bundle ID**（此处以 com.mpaas.demo 为例），点击 **下载配置** 下载配置文件。



3.6.3 在 Xcode 创建工程

在本节您将创建创建一个显示 **Hello World!** 应用。

点击 [视频教程](#)，查看对应的视频教程。

创建工程

1. 打开 Xcode。
2. 点击 **Edit** > **mPaaS** > **新建 mPaaS 工程**，输入 **项目名称** 和 **项目路径**。在此处项目名称以

MyH5Application 为例。



3. 导入从控制台下载到的 **配置文件**。Xcode 会自动解析并填充。



4. 选择 App 布局。



5. 选择 **H5容器&离线包** 项目组件。



6. 工程创建成功。



7. 点击 OK。

恢复 Xcode 签名。

说明：此步骤非必须步骤。当您发现在编译工程后模拟器无响应时，可恢复 Xcode 签名后再次尝试。



9. 点击 **Run**，工程能够编译通过并且在模拟器中进行安装。至此，您已完成创建工程。



3.6.4 使用 H5 容器

初始化容器

为了使用 Nebula 容器，您需要在程序启动完成后进行初始化。

在

MyH5Application/MPaaS/Targets/MyH5Application/APMobileFramework/DTFrameworkInterface+MyH5Application.m 中，DTFrameworkInterface 的 Category 的 beforeDidFinishLaunchingWithOptions 方法中，调用以下接口进行初始化。

示例如下：

```
- (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // 初始化容器
    [MPNebulaAdapterInterface initNebula];
}
```

在应用内打开一个在线网页

在客户端代码中添加完容器初始化逻辑后，就可以调用 Nebula 容器提供的接口，启动一个在线网页。

点击 [视频教程](#)，查看对应的视频教程。

在 MyH5Application/Sources/DemoViewController.m 中添加代码。
添加一个按钮，点击按钮调用接口打开在线网页。

```
#import "DemoViewController.h"

@interface DemoViewController ()

@end

@implementation DemoViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.

    UIButton *button = [UIButton buttonWithType:UIButtonTypeCustom];
    button.frame = CGRectMake(30, 150, [UIScreen mainScreen].bounds.size.width-60, 44);
    button.backgroundColor = [UIColor blueColor];
    [button setTitle:@"在线 URL" forState:UIControlStateNormal];
    [button addTarget:self action:@selector(openOnline) forControlEvents:UIControlEventTouchUpInside];
    [self.view addSubview:button];
}

- (void)openOnline
{
    [[MPNebulaAdapterInterface sharedInstance]
```

```
startH5ViewControllerWithParams:@{@"url":@"https://tech.antfin.com"};
}

@end
```

2. 点击按钮后即可在应用内打开金融科技官网首页。至此，您已完成 **在应用内打开一个在线网页**。



前端调用 Native 接口

在开发前端页面时，可以通过 Nebula 容器提供的 bridge，通过 JSAPI 的方式与 Native 进行通信，获取 Native 处理的相关信息或数据。Nebula 容器内预置了部分基础的 JSAPI 能力（详情参见 [链接](#)），您可以在 H5 页面的 js 文件中，直接通过 AlipayJSBridge.call 的方式进行调用。示例如下：

```
AlipayJSBridge.call('alert', {
  title: '原生 Alert Dialog',
  message: '这是一个来自原生的 Alert Dialog',
  button: '确定'
}, function (e) {
  alert("点击了按钮");
});
```

说明： <https://mcube-prod.oss-cn-hangzhou.aliyuncs.com/570DA89281533->

default/80000000/1.0.0.1_all/nebula/fallback/h5_to_native.html 是已经写好的前端页面，您可以调用此页面以体验前端调用 Native 接口的功能。

点击 [视频教程](#)，查看对应的视频教程。

在 MyH5Application/Sources/DemoViewController.m 中添加代码。
添加一个按钮 button1。

```
UIButton *button1 = [UIButton buttonWithTypeCustom];
button1.frame = CGRectOffset(button.frame, 0, 80);
button1.backgroundColor = [UIColor blueColor];
[button1 setTitle:@"前端调用 native" forState:UIControlStateNormal];
[button1 addTarget:self action:@selector(openJsApi) forControlEvents:UIControlEventTouchUpInside];
[self.view addSubview:button1];
```

在 MyH5Application/Sources/DemoViewController.m 中，给 button1 添加实现代码。如下所示：

```
- (void)openJsApi
{
[[MPNebulaAdapterInterface sharedInstance] startH5ViewControllerWithParams:@{@"url":@"https://mcube-prod.oss-cn-hangzhou.aliyuncs.com/570DA89281533-default/80000000/1.0.0.1_all/nebula/fallback/h5_to_native.html"}];
}
```

3. 编译工程后，在手机上安装应用。打开应用后界面如下所示。



4. 点击按钮后即可打开前端页面，点击按钮 **显示原生Alert Dialog**，会弹出原生的警示框，警示框的标题是 **原生 Alert Dialog**，消息框的内容是 **这是一个来自原生的 Alert Dialog**；点击警示框的 **确定** 按钮，会再弹出一个无标题警示框，内容是 **点击了按钮**。说明接口调用成功。至此，您已完成 **前端调用 Native 接口**。



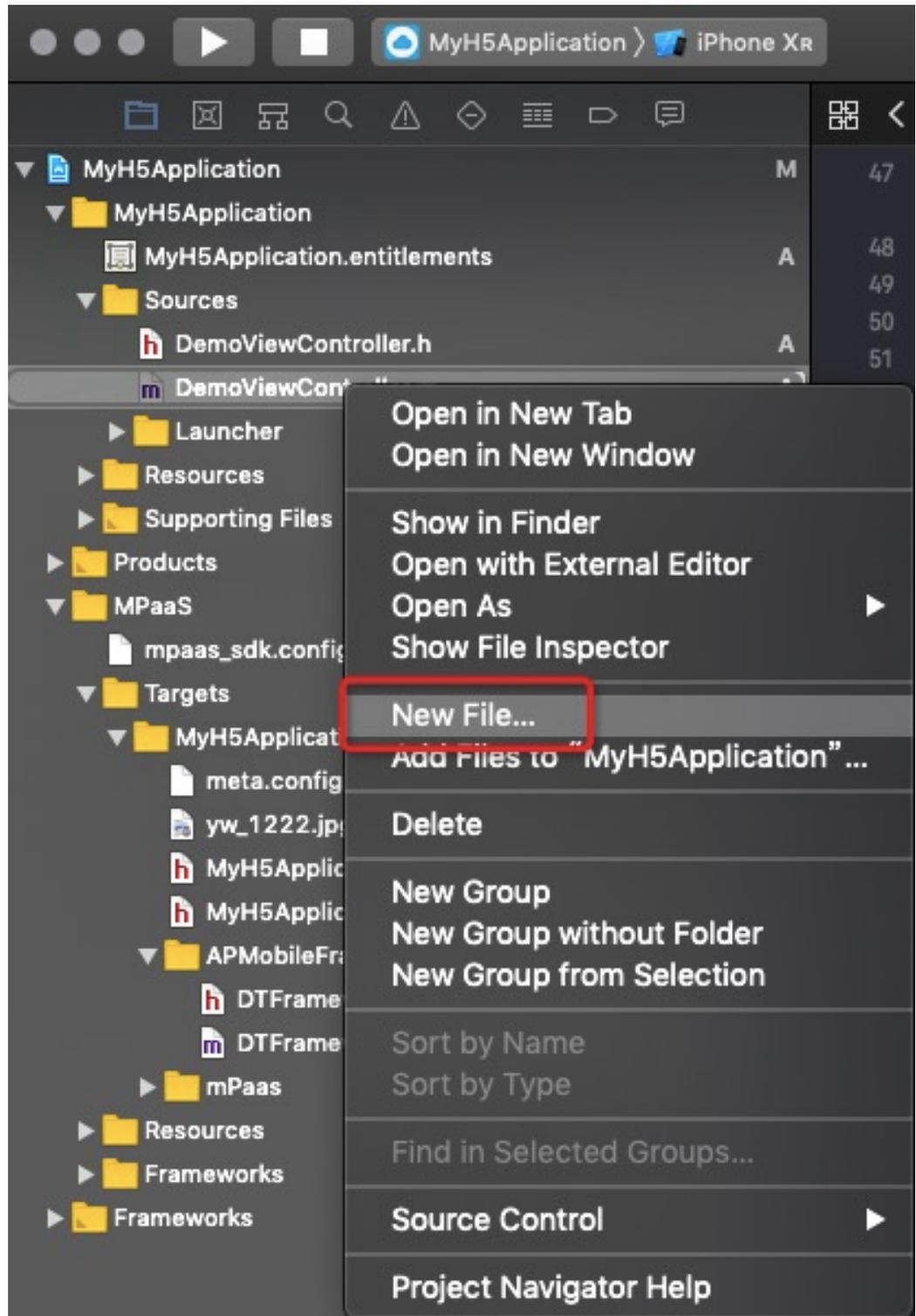
前端调用自定义 JSAPI

除了 Nebula 容器预置的基础 JSAPI 能力外，您还可以根据需要根据以下方式自定义一个 JSAPI。

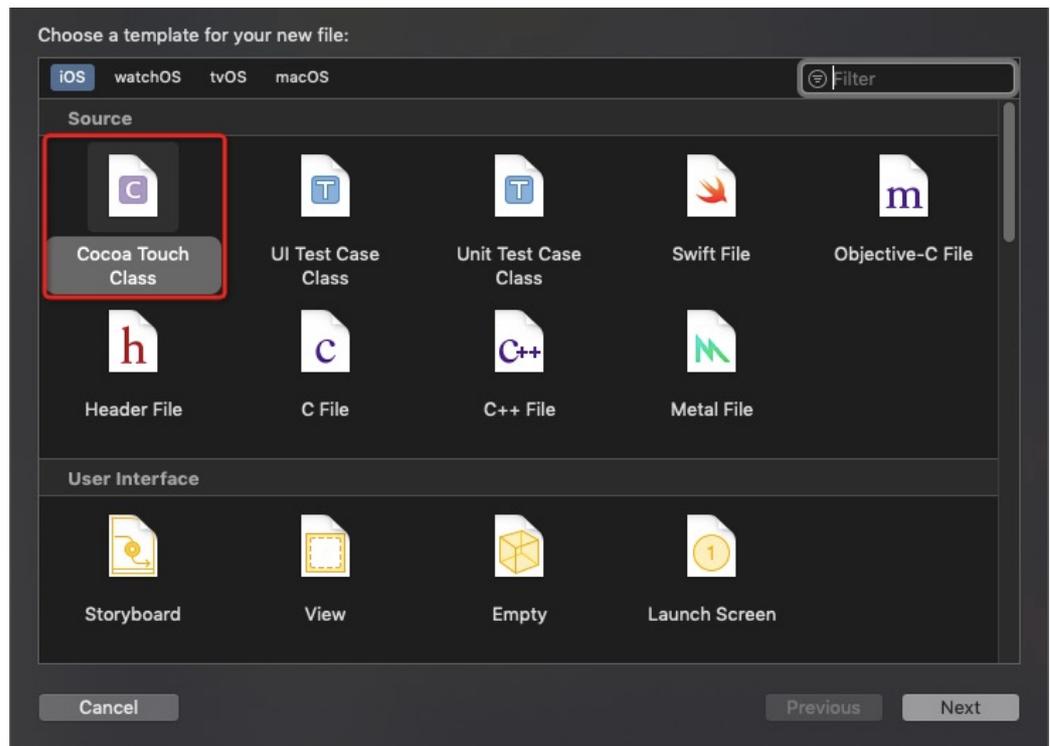
点击 [视频教程](#)，查看对应的视频教程。

创建 JSAPI 类。

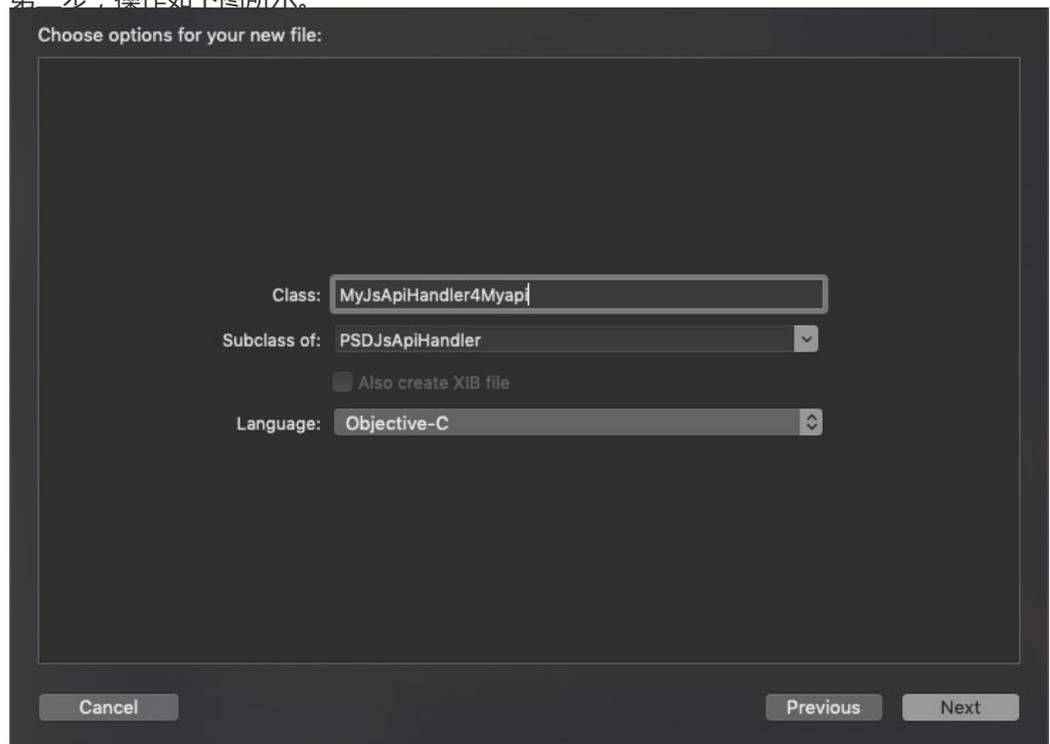
- 第一步，操作如下图所示。



◦ 第二步，操作如下图所示。



- 第三步，操作如下图所示。



- 第四步，添加代码。

```
@implementation MyJsApiHandler4Myapi
```

```
- (void)handler:(NSDictionary *)data context:(PSDContext *)context  
callback:(PSDJsApiResponseCallbackBlock)callback
```

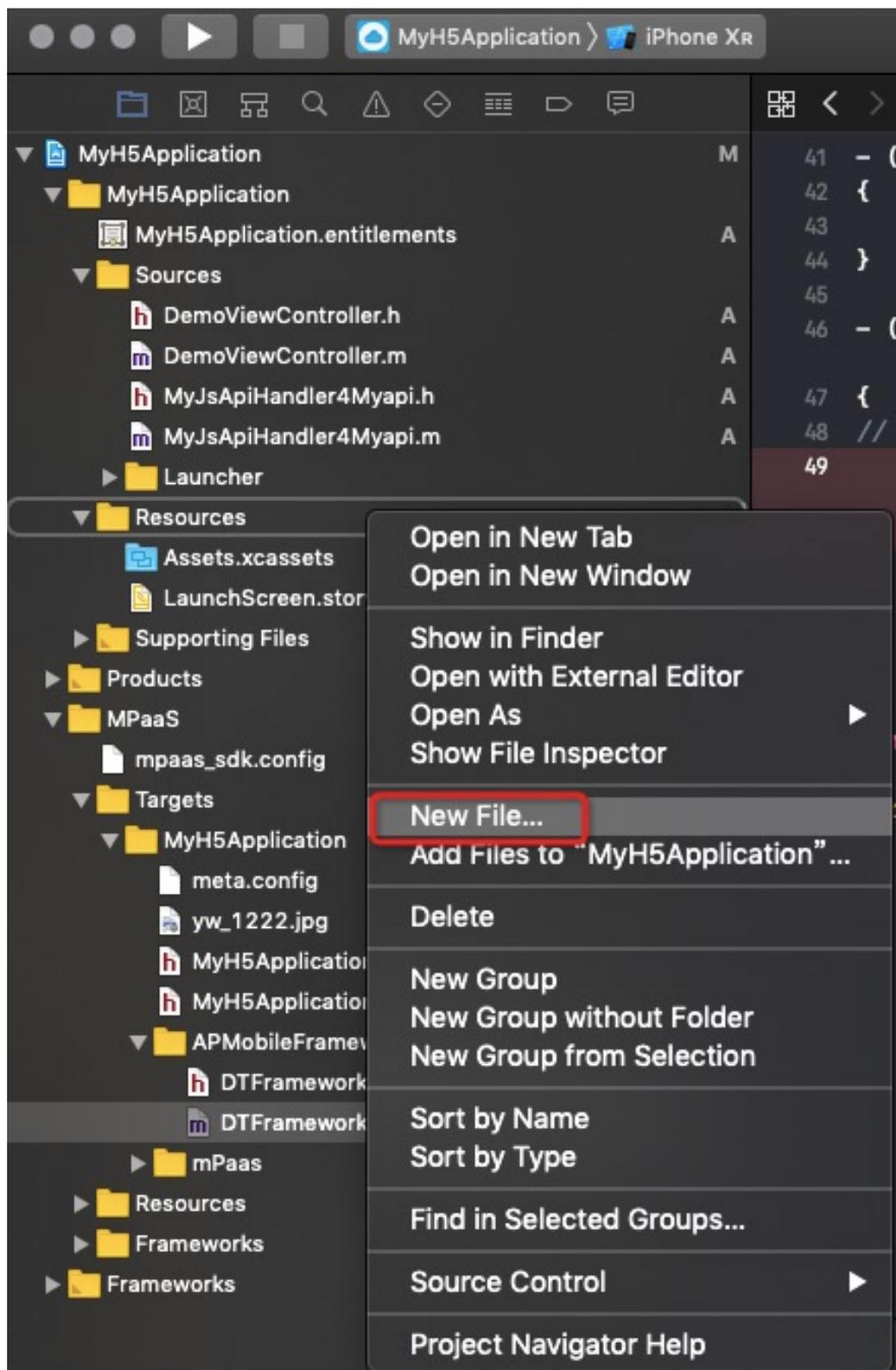
```
{
[super handler:data context:context callback:callback];

NSString *userId = @"admin";
if ([userId length] > 0) {
callback(@{"success":@YES, @"userId":userId});
} else {
callback(@{"success":@NO});
}
}

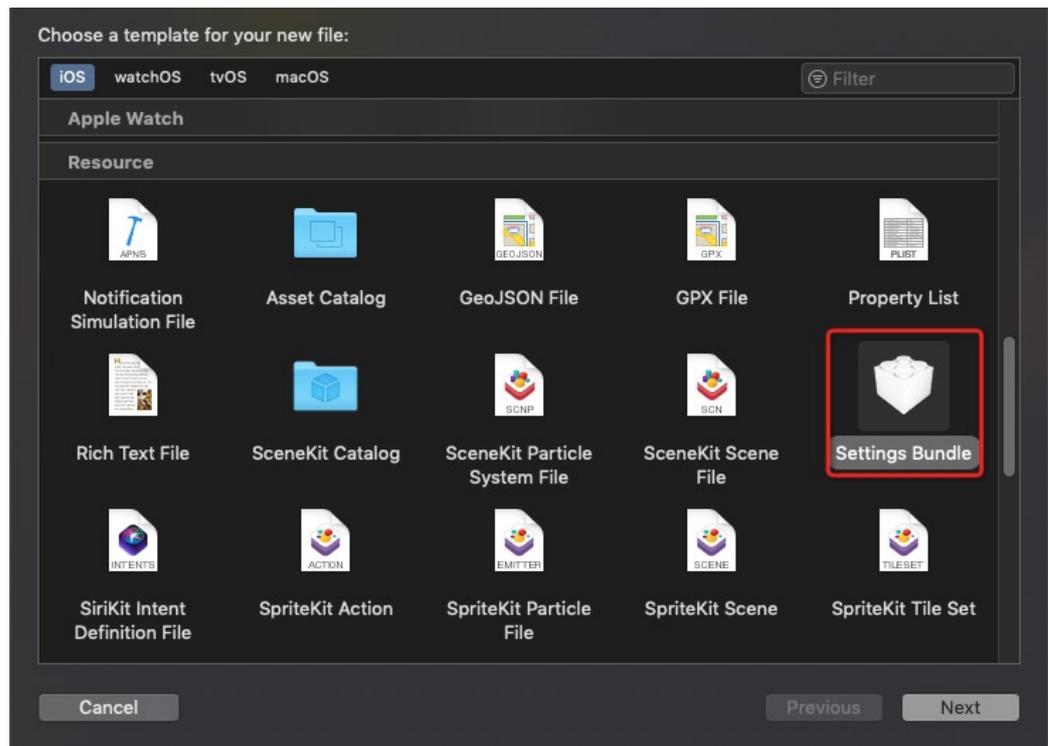
@end
```

2. 向 Nebula 容器注入您自定义的 JSAPI。

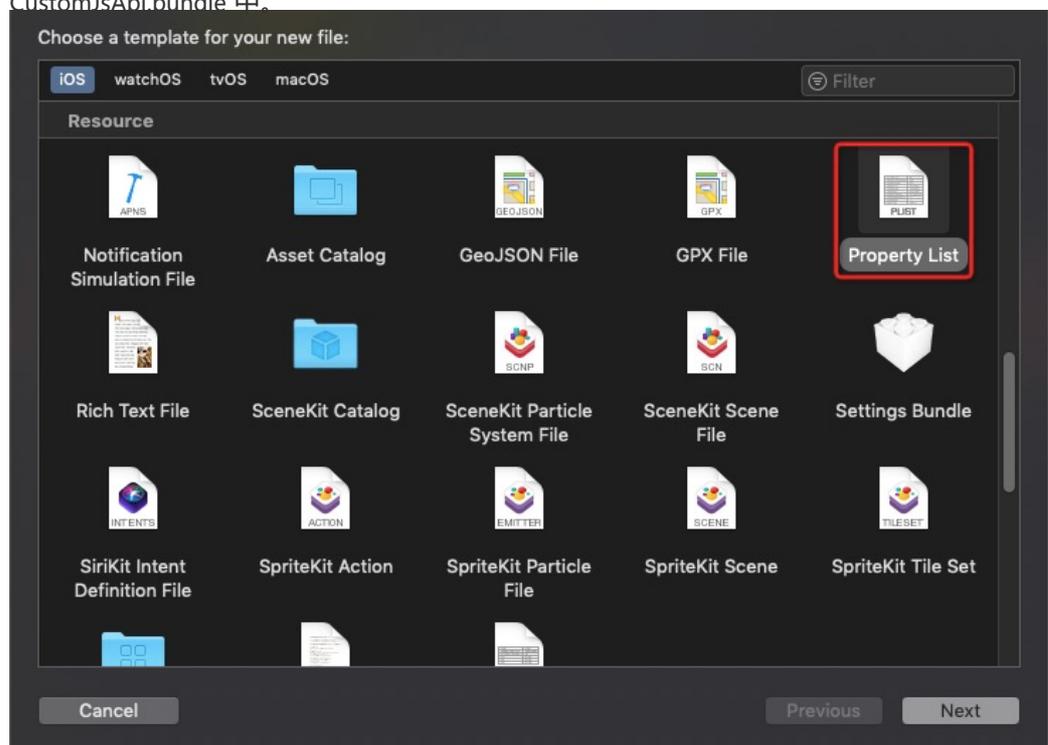
- 在 MyH5Application/MyH5Application/Resources 下新建一个 bundle。

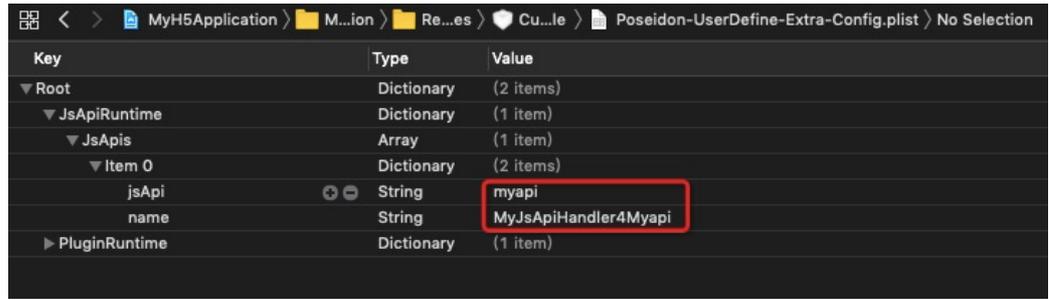


。按下图所示进行操作。



- 将 bundle 重命名为 CustomJsApi.bundle，并将其内容清空。
- 在 MyH5Application/MyH5Application/Resources 中新建一个 plist 文件，重命名为 Poseidon-UserDefine-Extra-Config.plist。在 JsApis 数组注册上一步中创建的 JsApi 类。注册的 JsApi 是一个字典 Dictionary 类型，其中 jsApi 表示在 H5 页面中调用的 JsApi 接口名 myapi；name 表示上一步创建的类名 MyJsApiHandler4Myapi。将 plist 文件拖拽到 CustomJsApi.bundle 中。





Key	Type	Value
Root	Dictionary	(2 items)
JsApiRuntime	Dictionary	(1 item)
JsApis	Array	(1 item)
Item 0	Dictionary	(2 items)
jsApi	String	myapi
name	String	MyJsApiHandler4Myapi
PluginRuntime	Dictionary	(1 item)

向容器注册自定义 JsApi 的路径。在

MyH5Application/MPaaS/Targets/MyH5Application/APMobileFramework/DTFrameworkInterface+MyH5Application.m 中，使用下面的接口，初始化容器。在采用自定义 JSAPI 的时候，需要采用以下方法，通过 plist 将自定义的类注入容器，将自定义的路径 NSS string 传给初始化方法。

```
- (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
// [MPNebulaAdapterInterface initNebula];
NSString *pluginsJsapisPath = [[NSBundle mainBundle] pathForResource:[NSString stringWithFormat:@"CustomJsApi.bundle/Poseidon-UserDefine-Extra-Config.plist"] ofType:nil];
[MPNebulaAdapterInterface
initNebulaWithCustomPresetApplistPath:@"customPresetAppPackagePath:@"customPluginsJsapisPath:pluginsJsapisPath];
}
```

在 MyH5Application/Sources/DemoViewController.m 中添加代码，新增按钮 button2。

```
UIButton *button2 = [UIButton buttonWithType:UIButtonTypeCustom];
button2.frame = CGRectOffset(button1.frame, 0, 80);
button2.backgroundColor = [UIColor blueColor];
[button2 setTitle:@"前端调用自定义 JSAPI" forState:UIControlStateNormal];
[button2 addTarget:self action:@selector(openCusJsApi) forControlEvents:UIControlEventTouchUpInside];
[self.view addSubview:button2];
```

在 MyH5Application/Sources/DemoViewController.m 中添加代码，定义 openCusJsApi。实现代码如下所示：

```
- (void)openCusJsApi
{
[[MPNebulaAdapterInterface sharedInstance] startH5ViewControllerWithParams:@{@"url":@"https://mcube-prod.oss-cn-hangzhou.aliyuncs.com/570DA89281533-default/80000001/1.0.0.1_all/nebula/fallback/custom_jsapi.html"}];
}
```

说明： https://mcube-prod.oss-cn-hangzhou.aliyuncs.com/570DA89281533-default/80000001/1.0.0.1_all/nebula/fallback/custom_jsapi.html 是已经写好的前端页面，您可以调用此页面以体验前端调用自定义 JSAPI 接口的功能。在该前端页面中，通过以下方法调用该自定义 JSAPI。示例如下所示。

```
AlipayJSBridge.call('myapi', {  
  param1: 'JsParam1',  
  param2: 'JsParam2'  
}, function (result) {  
  alert(JSON.stringify(result));  
});
```

6. 编译工程后，在手机上安装应用。打开应用后界面如下所示。



7. 点击 **前端调用自定义 JSAPI** 按钮后即可打开前端页面，再点击按钮 **自定义JSAPI**，会打开包含了一个按钮 **自定义JSAPI** 的前端页面。点击该 **自定义JSAPI** 按钮，会再弹出一个无标题警示框，内容按照自定义 API 定义的功能处理了的前端调用时传入的参数。至此，您已完成 **前端调用 自定义JSAPI**



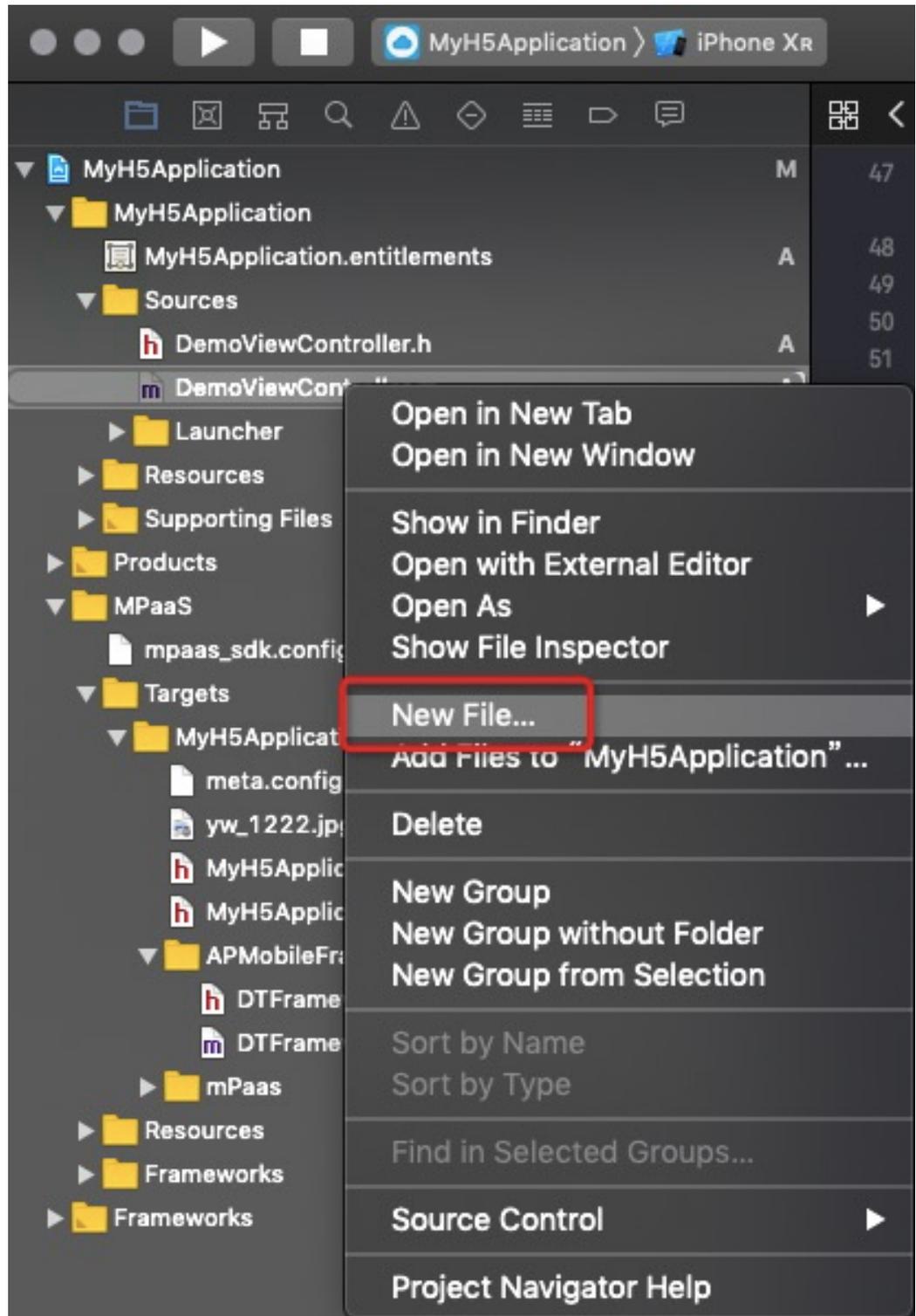
自定义H5页面的 TitleBar

H5 页面导航栏样式为白底黑字蓝按钮，您可以通过自定义一个 Plugin，来修改导航栏样式。

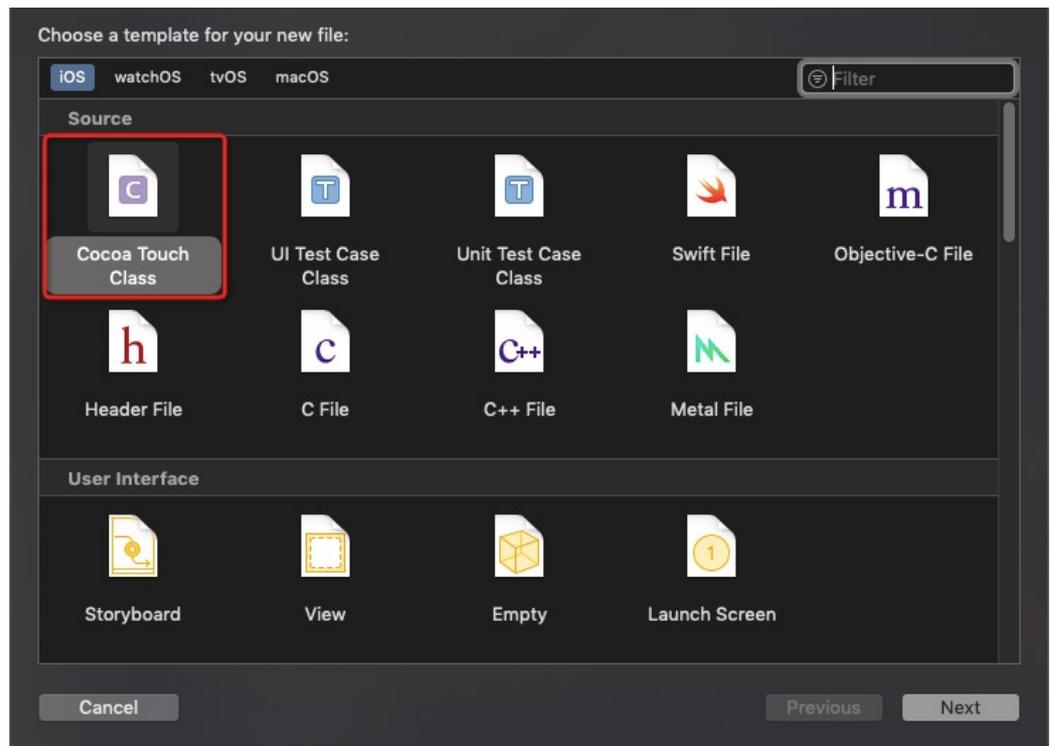
点击 [视频教程](#)，查看对应的视频教程。

创建 Plugin 类。

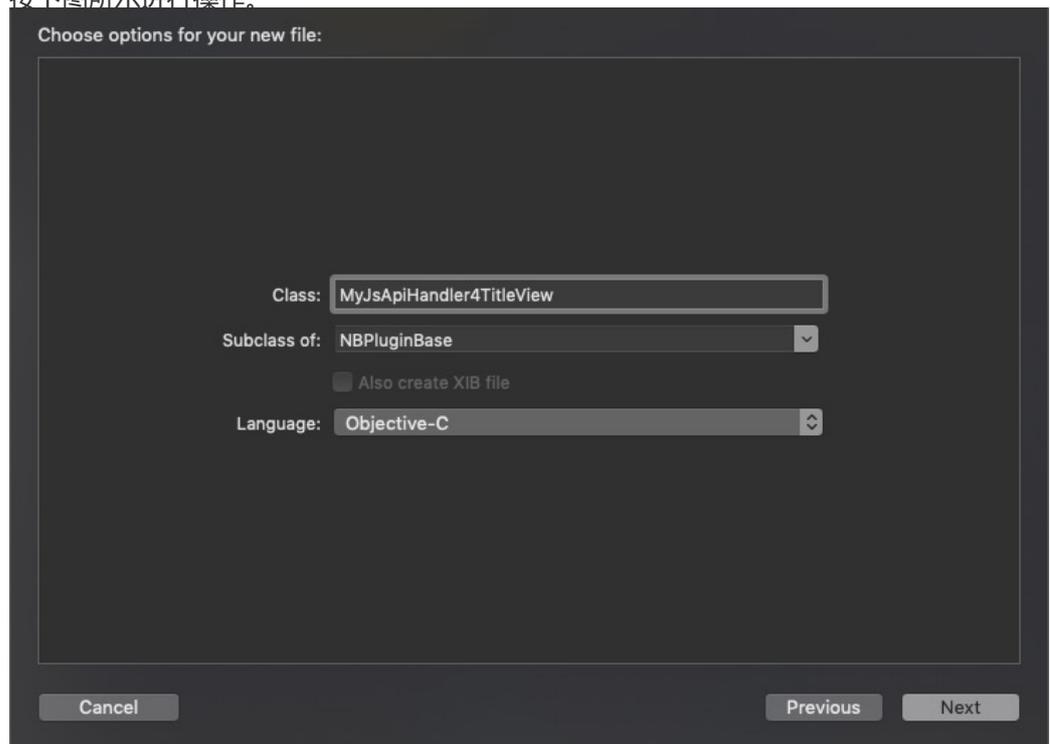
- 按下图所示进行操作。



- 按下图所示进行操作。



- 。按下图所示进行操作。



添加代码。

```
- (void)pluginDidLoad
{
    self.scope = kPSDScope_Scene;
}
```

```

// -- 返回区域
[self.target addEventListener:kNBEvent_Scene_NavigationItem_Left_Back_Create_Before withListener:self
useCapture:NO];

[super pluginDidLoad];
}

- (void)handleEvent:(PSDEvent *)event
{
[super handleEvent:event];

if ([kNBEvent_Scene_NavigationItem_Left_Back_Create_Before isEqualToString:event.eventType]) {
// 在默认返回按钮基础上，修改样式
NSArray *leftBarButtonItems = event.context.currentViewController.navigationItem.leftBarButtonItems;
if ([leftBarButtonItems count] == 1) {
if (leftBarButtonItems[0] && [leftBarButtonItems[0] isKindOfClass:[AUBarButtonItem class]]) {
// 在默认返回按钮基础上，修改返回箭头和文案颜色
AUBarButtonItem *backItem = leftBarButtonItems[0];
backItem.backButtonColor = [UIColor redColor];
backItem.titleColor = [UIColor redColor];
}
}
[event preventDefault];
[event stopPropagation];
}
}

- (int)priority
{
return PSDPluginPriority_High;
}

```

2. 向 Nebula 容器注入您自定义的 JSAPI。

打开 Poseidon-UserDefine-Extra-Config.plist 文件，在 Plugins 数组注册上一步中创建的 MPJsApiHandler4TitleView 类。注册的 Plugins 是一个字典 Dictionary 类型，其中 name 表示上一步创建的类名 MyJsApiHandler4TitleView。Scope 表示处理范围是页面（scene）。

Key	Type	Value
▼ Root	Dictionary	(2 items)
▶ JsApiRuntime	Dictionary	(1 item)
▼ PluginRuntime	Dictionary	(1 item)
▼ Plugins	Array	(1 item)
▼ Item 0	Dictionary	(3 items)
name	String	MPJsApiHandler4TitleView
scope	String	scene
▶ events	Array	(1 item)

向容器注册自定义 plugin 的路径。在

MyH5Application/MPaaS/Targets/MyH5Application/APMobileFramework/DTFrameworkInterface+MyH5Application.m 中，使用下面的接口，初始化容器。在采用自定义 JSAPI 的时候，需要采用以下方法，通过 plist 将自定义的 plugin 类注入容器。

说明：在上一步调用自定义 JSAPI 时已完成注册，可跳过本步骤。

```

- (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
// [MPNebulaAdapterInterface initNebula];
NSString *pluginsJsapisPath = [[NSBundle mainBundle] pathForResource:[NSString stringWithFormat:@"CustomJsApi.bundle/Poseidon-UserDefine-Extra-Config.plist"] ofType:nil];
[MPNebulaAdapterInterface
initNebulaWithCustomPresetApplistPath:@"customPresetAppPackagePath:@"customPluginsJsapisPath:pluginsJsapisPath];
}

```

在 MyH5Application/Sources/DemoViewController.m 中添加代码，新增按钮 button3。

```

UIButton *button3 = [UIButton buttonWithType:UIButtonTypeCustom];
button3.frame = CGRectOffset(button2.frame, 0, 80);
button3.backgroundColor = [UIColor blueColor];
[button3 setTitle:@"自定义导航栏" forState:UIControlStateNormal];
[button3 addTarget:self action:@selector(customNavigatorBar)
forControlEvents:UIControlEventTouchUpInside];
[self.view addSubview:button3];

```

在 MyH5Application/Sources/DemoViewController.m 中添加代码，定义 customNavigatorBar。实现代码如下所示。

```

- (void)customNavigatorBar
{
[[MPNebulaAdapterInterface sharedInstance]
startH5ViewControllerWithParams:@{@"url":@"https://tech.antfin.com"}];
}

```

6. 编译工程后，在手机上安装应用。打开应用后界面如下所示。



7. 点击 **自定义导航** 按钮后即可打开金融科技官网，此时返回按钮已经根据自定义调整样式。至此，您已完成 **自定义导航**。



3.6.5 使用 H5 离线包

参考 在 Xcode 创建工程 ，创建新工程，我们基于此工程，使用 H5 离线包。

发布离线包

首先，在您使用 H5 离线包之前，您需要先准备一个前端 App 的 zip 包。如果您没有自己的前端离线包，您可以下载我们为您准备好的 [示例离线包](#)。

点击 [视频教程](#)，查看对应的视频教程。

1. 您需要先在控制台的应用中配置离线包信息，参考 [配置离线包](#)。
2. 您需要生成您自己前端 App 的离线包（或者使用我们的示例离线包），参考 [生成离线包](#)。
3. 您需要在控制台上创建该离线包并上传，参考 [创建离线包](#)。
4. 您需要将配置好的离线包发布到您的客户端 App 中，参考 [发布离线包](#)。

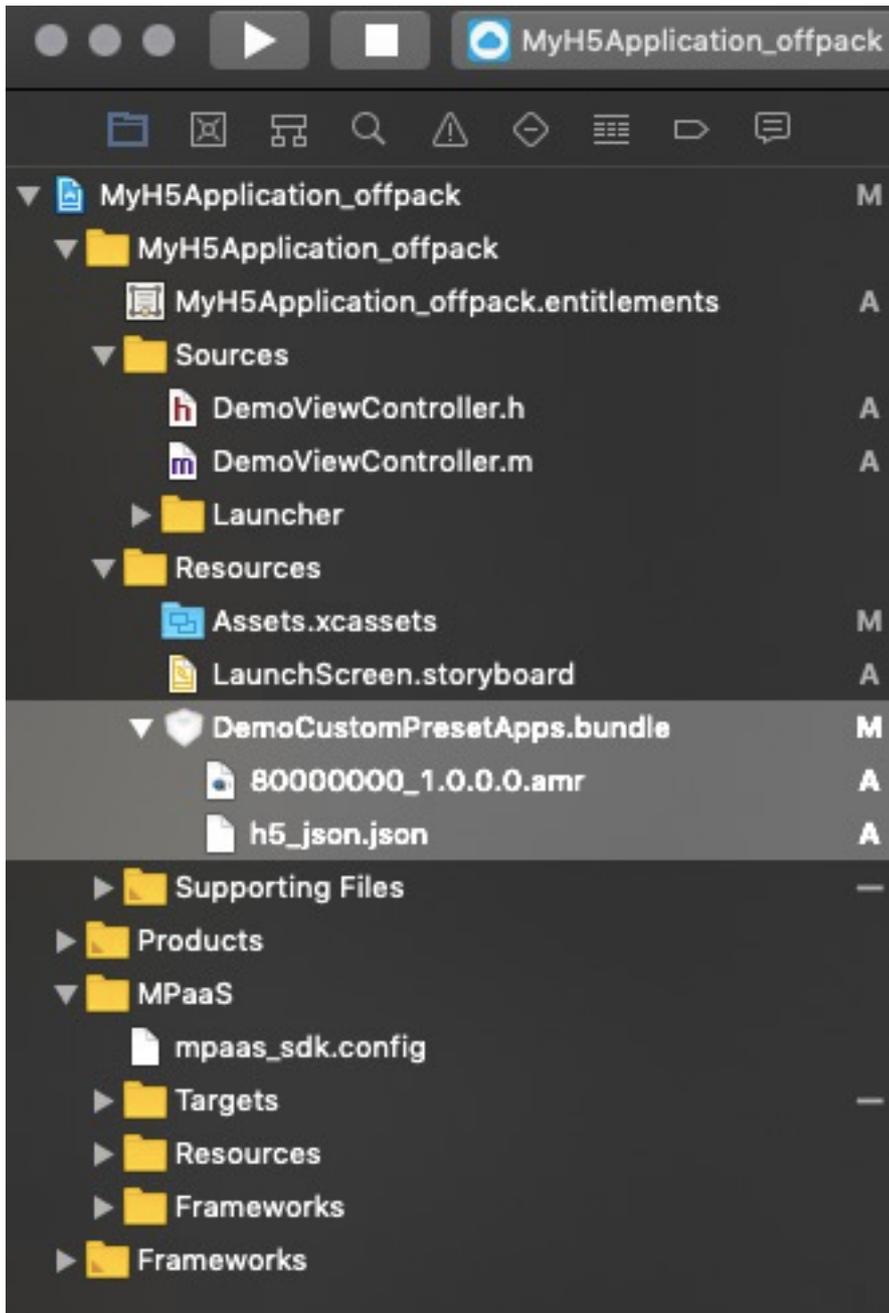
预置离线包

点击 [视频教程](#)，查看对应的视频教程。

1. 在控制台上下载离线包 AMR 文件，以及离线包配置文件到本地。



2. 在 MyH5Application_offpack/Resources 下创建 DemoCustomPresetApps.bundle，将 bundle 下的内容清空，并将下载到的离线包 AMR 文件和离线包配置文件添加到此 bundle 中。



向容器注册预置的离线包。将包含预置离线包信息的 plist 文件和离线包的路径分别赋给 presetApplistPath 和 appPackagePath，通过 initNebulaWithCustomPresetApplistPath 进行初始化。至此，您已经完成 **预置离线包**。

```

- (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    NSString *presetApplistPath = [[NSBundle mainBundle] pathForResource:[NSString stringWithFormat:@"DemoCustomPresetApps.bundle/h5_json.json"] ofType:nil];
    NSString *appPackagePath = [[NSBundle mainBundle] pathForResource:[NSString stringWithFormat:@"DemoCustomPresetApps.bundle"] ofType:nil];
    [MPNebulaAdapterInterface initNebulaWithCustomPresetApplistPath:presetApplistPath

```

```
customPresetAppPackagePath:appPackagePath customPluginsJsapisPath:@""];
}
```

启动离线包

点击 [视频教程](#)，查看对应的视频教程。

在 MyH5Application_offpack/Sources/DemoViewController.m 中添加代码。添加按钮 button4，点击该按钮以调用接口打开离线包页面。

```
UIButton *button4 = [UIButton buttonWithType:UIButtonTypeCustom];
button4.frame = CGRectMake(30, 150, [UIScreen mainScreen].bounds.size.width-60, 44);
button4.backgroundColor = [UIColor blueColor];
[button4 setTitle:@"打开离线包页面" forState:UIControlStateNormal];
[button4 addTarget:self action:@selector(startOffPack) forControlEvents:UIControlEventTouchUpInside];
[self.view addSubview:button4];
```

在 MyH5Application_offpack/Sources/DemoViewController.m 中，给 button4 添加实现代码启动离线包。其中传入的参数 80000000 为离线包的 APPID。具体代码如下所示。

```
- (void)startOffPack
{
[[MPNebulaAdapterInterface sharedInstance]
startH5ViewControllerWithNebulaApp:@{@"appId":@"80000000"}];
}
```

编译工程后，在手机上安装应用。打开应用后界面如下所示。



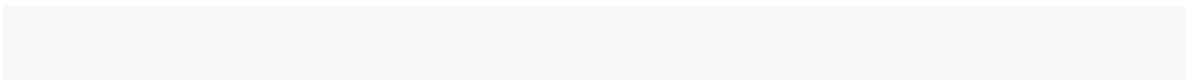
4. 点击按钮 **打开离线包页面**，即可打开离线包中预置的网页，如下图所示。至此，您已完成 **启动离线包**。



更新离线包

点击 [视频教程](#)，查看对应的视频教程。

1. 在 MyH5Application_offpack/Sources/DemoViewController.m 中添加代码。添加按钮 button5，点击按钮调用接口打开离线包页面。



```
UIButton *button5 = [UIButton buttonWithType:UIButtonTypeCustom];
button5.frame = CGRectOffset(button4.frame, 0, 80);
button5.backgroundColor = [UIColor blueColor];
[button5 setTitle:@"更新离线包" forState:UIControlStateNormal];
[button5 addTarget:self action:@selector(updateOffPack) forControlEvents:UIControlEventTouchUpInside];
[self.view addSubview:button5];
```

在 MyH5Application_offpack/Sources/DemoViewController.m 中，给 button5 添加实现代码以更新离线包。

```
- (void)updateOffPack
{
    [[MPNebulaAdapterInterface sharedInstance] requestNebulaAppsWithParams:@{@"80000000":@"*"}
    finish:^(NSDictionary *data, NSError *error) {
        NSString *result = @"后台无新包发布";
        if (!error && [data[@"data"] count] > 0) {
            result = [NSString stringWithFormat:@"%@", data[@"data"]];
        }

        dispatch_async(dispatch_get_main_queue(), ^{
            UIAlertView* alertView = [[UIAlertView alloc] initWithTitle:@"离线包已更新" message:result delegate:nil
            cancelButtonTitle:@"OK" otherButtonTitles:nil, nil];
            [alertView show];
        });
    });
}
```

3. 编译工程后，在手机上安装应用。打开应用后界面如下所示。



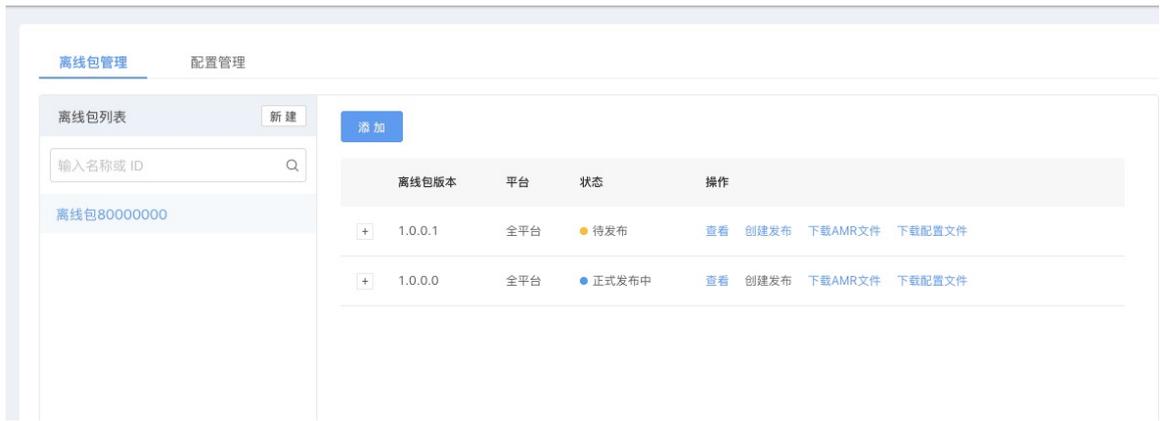
4. 此时点击 **打开离线包页面** 按钮，会显示以下页面。



5. 点击 **更新离线包** 按钮，则会弹出 toast 提示 **后台无新包发布**。



6. 更新离线包文件，在控制台中上传并发布。在本例中更新的内容是将按钮显示文字更新为 **更新 20190826**。



说明：在控制台上传新版本离线包以更新客户端离线包时，需要填写支持的客户端范围。此范围由最低版本和最高版本两个版本号确定。只有在此版本范围内的客户端，才能够得到推送的新版本离线包。客户端版本信息可以在客户端工程的 info.plist 文件的 product version 获得。

基本信息

* 资源包类型 [?]: 全局资源包 普通资源包

* 版本 [?]:

* 客户端范围: iOS 最低版本: 最高版本:

Android 最低版本: 最高版本:

* 文件 [?]:

7. 在应用中点击 **更新离线包** 按钮，此时应弹出 toast 提示 **离线包已更新**，如下图所示。



此时点击 打开 **离线包页面** 按钮，会显示以下页面。



至此，您已经完成 **更新离线包**。

3.7 参考说明

3.7.1 定制 H5 页面导航栏

前置条件

在定制 H5 页面的导航栏样式前，您需要确保已经了解框架导航栏和 H5 容器相关知识，主要包括：

- iOS 定制导航栏
- 自定义 JSAPI
- 自定义插件

设置所有 H5 页面导航栏默认样式

在应用主题的基础上，若您需要指定所有 H5 页面统一的样式，可以通过监听 Nebula 容器提供的事件机制进行定制。

容器支持的事件，参见头文件 NebulaSDK/NBDefine.h：

```

125 #define kNBEvt_Scene_TitleView_All @"scene.titleview.*"
126
127 // navitaion item btn
128
129 #define kNBEvt_Scene_NavigationItem_Left_Back_Create_Before @"scene.navigationItem.left.back.create.before" // 监听该事件，可以调用
    preventDefault阻止默认行为（创建默认的返回按钮），若要自定义返回按钮，则设置其`customView`
130 #define kNBEvt_Scene_NavigationItem_Left_Back_Create_After @"scene.navigationItem.left.back.create.after" // 监听该事件，可以获得返
    回按钮，并设置它的属性
131 #define kNBEvt_Scene_NavigationItem_Left_Back_Click @"scene.navigationItem.left.back.click" // 监听该事件，可以调用
    preventDefault阻止默认行为（默认是返回行为），并自己实现返回逻辑
132 #define kNBEvt_Scene_NavigationItem_Left_Back_All @"scene.navigationItem.left.back.*"
133
134
135 #define kNBEvt_Scene_NavigationItem_Left_Close_Create_Before @"scene.navigationItem.left.close.create.before" // 监听该事件，可以调用
    preventDefault阻止默认行为（创建默认的关闭按钮），若要自定义关闭按钮，则设置其`customView`
136 #define kNBEvt_Scene_NavigationItem_Left_Close_Create_After @"scene.navigationItem.left.close.create.after" // 监听该事件，可以获取关
    闭按钮，并设置其属性
137 #define kNBEvt_Scene_NavigationItem_Left_Close_Click @"scene.navigationItem.left.close.click" // 监听该事件，可以调用
    preventDefault阻止默认行为（默认是关闭行为），并自己实现关闭行为
138 #define kNBEvt_Scene_NavigationItem_Left_Close_All @"scene.navigationItem.left.close.*"
139
140
141 #define kNBEvt_Scene_NavigationItem_Right_Setting_Create_Before @"scene.navigationItem.right.setting.create.before" // 监听该事件，
    可以调用preventDefault阻止默认行为（默认是创建设置按钮），若要自定义设置按钮，则设置其`customView`，自定义的customView要实现协议
    NBSettingButtonProtocol
142 #define kNBEvt_Scene_NavigationItem_Right_Setting_Create_After @"scene.navigationItem.right.setting.create.after" // 监听该事件，可

```

自定义插件 监听需要的事件进行定制化：

```

@implementation MPPlugin4TitleView

- (void)pluginDidLoad
{
self.scope = kPSDScope_Scene;
// -- 返回区域
[self.target addEventListener:kNBEvt_Scene_NavigationItem_Left_Back_Create_Before withListener:self
useCapture:NO];
[self.target addEventListener:kNBEvt_Scene_NavigationItem_Left_Back_Create_After withListener:self
useCapture:NO];
[self.target addEventListener:kNBEvt_Scene_NavigationItem_Left_Close_Create_Before withListener:self
useCapture:NO];
[self.target addEventListener:kNBEvt_Scene_NavigationItem_Left_Close_Create_After withListener:self
useCapture:NO];

// -- 标题区域
[self.target addEventListener:kNBEvt_Scene_TitleView_Create_Before withListener:self useCapture:NO];

```

```
[self.target addEventListener:kNBEvent_Scene_TitleView_Create_After withListener:self useCapture:NO];

// -- 控制按钮区域
[self.target addEventListener:kNBEvent_Scene_NavigationItem_Right_Setting_Create_Before withListener:self
useCapture:NO];
[self.target addEventListener:kNBEvent_Scene_NavigationItem_Right_Setting_Create_After withListener:self
useCapture:NO];
[self.target addEventListener:kNBEvent_Scene_NavigationItem_Right_SubSetting_Create_After withListener:self
useCapture:NO];
[self.target addEventListener:kNBEvent_Scene_NavigationItem_Right_Setting_Change withListener:self
useCapture:NO];

// -- 进度条
[self.target addEventListener:kNBEvent_Scene_ProgressView_Create_Before withListener:self useCapture:NO];
[self.target addEventListener:kNBEvent_Scene_ProgressView_Create_After withListener:self useCapture:NO];

[super pluginDidLoad];
}
```

- 设置返回区域中返回按钮和关闭按钮的样式：

```
- (void)handleEvent:(PSDEvent *)event
{
[super handleEvent:event];

if ([kNBEvent_Scene_NavigationItem_Left_Back_Create_Before isEqualToString:event.eventType]) {
[event preventDefault];

event.context.currentViewController.navigationItem.leftBarButtonItem = [[UIBarButtonItem alloc] initWithTitle:@"取消" style:UIBarButtonItemStylePlain target:self action:@selector(onClickBack)];
}else if ([kNBEvent_Scene_NavigationItem_Left_Back_Create_After isEqualToString:event.eventType]){
// 修改返回按钮样式
NSArray *leftBarButtonItems = event.context.currentViewController.navigationItem.leftBarButtonItems;
if ([leftBarButtonItems count] == 1) {
if ([leftBarButtonItems[0] isKindOfClass:[AUIBarButtonItem class]]) {
// 在默认返回按钮基础上，修改返回箭头和文案颜色
AUIBarButtonItem *backItem = leftBarButtonItems[0];
backItem.backButtonColor = [UIColor greenColor];
backItem.titleColor = [UIColor colorWithHexString:@"#00ff00"];

// 隐藏返回箭头
// backItem.hideBackButtonImage = YES;

// 隐藏返回文案：文案设置为透明，保留返回按钮 s 点击区域
// backItem.titleColor = [UIColor clearColor];
}
}
}else if ([kNBEvent_Scene_NavigationItem_Left_Close_Create_Before isEqualToString:event.eventType]){
// // 隐藏关闭按钮
// [event preventDefault];
// UINavigationControllerLeftCloseEvent *itemEvent = (NINavigationControllerLeftCloseEvent *)event;
// UIButton *button = [UIButton buttonWithType:UIButtonTypeCustom];
// button.frame = CGRectMake(0, 0, 44, 44);
// button.backgroundColor = [UIColor greenColor];
// [button setTitle:@"Close" forState:UIControlStateNormal];
```

```
// itemEvent.customView = button;
}else if ([kNBEvent_Scene_NavigationItem_Left_Close_Create_After isEqualToString:event.eventType]){
// // 修改关闭按钮样式
// [event preventDefault];
// NBNavigationItemLeftCloseEvent *itemEvent = (NBNavigationItemLeftCloseEvent *)event;
// UIButton *closeButton = (UIButton *)itemEvent.customView;
// [closeButton setTitle:@"Close" forState:UIControlStateNormal];
// [closeButton setTitleColor:[UIColor greenColor] forState:UIControlStateNormal];
}
}
```

- 设置 H5 页面标题样式：

```
if ([kNBEvent_Scene_TitleView_Create_Before isEqualToString:event.eventType]) {
// 重写 TitleView 的样式
NBNavigationTitleViewEvent *e = (id)event;
[e preventDefault];

}else if ([kNBEvent_Scene_TitleView_Create_After isEqualToString:event.eventType]) {
// 更改已创建 TitleView 的样式
NBNavigationTitleViewEvent *e = (id)event;
[[e.titleView mainTitleLabel] setFont:[UIFont systemFontOfSize:16]];
[[e.titleView mainTitleLabel] setTitleColor:[UIColor greenColor]];
[e.titleView mainTitleLabel].lineBreakMode = NSLineBreakByTruncatingMiddle;
}
}
```

- 设置 OptionMenu 控制按钮的样式：

```
if ([kNBEvent_Scene_NavigationItem_Right_Setting_Create_After isEqualToString:event.eventType] ||
[kNBEvent_Scene_NavigationItem_Right_SubSetting_Create_After isEqualToString:event.eventType]) {
// 更改已创建 RightBarItem 的样式
NBNavigationItemRightSettingEvent *settingEvent = (id)event;
settingEvent.adjustsWidthToFitText = YES;
settingEvent.maxWidth = [UIScreen mainScreen].bounds.size.width / 3.0f;
UIButton *button = settingEvent.customView;
button.titleLabel.font = [UIFont systemFontOfSize:14.0f];
CGRect frame = CGRectMake(0, 0, 22, 22);
button.frame = frame;
[button setTitleColor:[UIColor blackColor] forState:UIControlStateNormal];
if (![CGSizeEqualToSize(button.bounds.size, frame.size)]) {
button.frame = frame;
}
}
}
```

- 设置 H5 页面加载时进度条的样式：

```
if([kNBEvent_Scene_ProgressView_Create_After isEqualToString:event.eventType]){
NBProgressViewEvent *progressEvent = (NBProgressViewEvent *)event;
id<NBProgressViewProtocol> progressView = progressEvent.progressView;
[progressView setProgressTintColor:[UIColor greenColor]];
}
}
```

定制某个 H5 页面导航栏样式

若您需要定制某个 H5 页面导航栏的样式，根据修改时机不同，提供的方法也分为两类：**页面加载前**与**页面打开后**。

页面加载前：在默认导航栏样式基础上定制导航栏，主要分为以下两步：

自定义启动参数：当前 H5 页面加载时自定义启动参数，指定定制方式：

- 从 H5 页面进入另一个 H5 页面，传递自定义启动参数的方法参见 `pushWindow` 打开新页面 和 `startApp` 启动其他应用。
- 从 Native 页面进入一个 H5 页面，传递自定义启动参数的方法参考如下代码：

```
#pragma mark 进入页面时修改，H5 需通过启动参数设置
- (void)gotoHideNavigator
{
    // 打开 H5 页面，隐藏导航栏
    [[MPNebulaAdapterInterface sharedInstance] startH5ViewControllerWithParams:@{@"url":
    @"https://tech.antfin.com", @"showTitleBar":@NO,@"transparentTitle":@"auto"}];
}

- (void)gotoShowNavigator
{
    // 打开 H5 页面，显示导航栏
    [[MPNebulaAdapterInterface sharedInstance] startH5ViewControllerWithParams:@{@"url":
    @"https://tech.antfin.com", @"showTitleBar":@YES}];
}

- (void)gotoTransparency
{
    // 打开 H5 页面，设置透明导航栏
    [[MPNebulaAdapterInterface sharedInstance] startH5ViewControllerWithParams:@{@"url":
    @"https://tech.antfin.com", @"transparentTitle":@"auto"}];
}

- (void)gotoUpdateBackgroundColor
{
    // 修改导航栏背景颜色
    [[MPNebulaAdapterInterface sharedInstance] startH5ViewControllerWithParams:@{@"url":
    @"https://tech.antfin.com", @"titleBarColor":@"16775138"}];
}

- (void)gotoUpdateStatusBarStyle
{
    // 修改状态栏颜色
    [[UIApplication sharedApplication] setStatusBarStyle:UIStatusBarStyleLightContent];
}

- (void)gotoUpdateBackTitleColor
{
    // 修改默认返回按钮文案颜色
    [[MPNebulaAdapterInterface sharedInstance] startH5ViewControllerWithParams:@{@"url":
    @"https://tech.antfin.com", @"backButtonColor":@"ff0000"}];
}
```

```

- (void)gotoUpdateTitleColor
{
// 修改标题颜色
[[MPNebulaAdapterInterface sharedInstance] startH5ViewControllerWithParams:@{@"url":
@"https://tech.antfin.com", @"titleColor":@"ff0000"}];
}

```

处理 H5 基类：在基类的 viewWillAppear 方法中，根据传入的启动参数调用 native 接口方法对导航栏样式进行修改：

```

- (void)viewWillAppear:(BOOL)animated
{
[super viewWillAppear:animated];

// 当前页面的 WebView
UIWebView *webView = (UIWebView *)self.psdContentView;
NSLog(@"[mpaas] webView: %@", webView);

// 当前页面的启动参数
NSDictionary *expandParams = self.psdScene.createParam.expandParams;
NSLog(@"[mpaas] expandParams: %@", expandParams);

if ([expandParams count] > 0) {
[self customNavigationBarWithParams:expandParams];
}
}

- (void)customNavigationBarWithParams:(NSDictionary *)expandParams
{
// 定制导航栏背景
NSString *titleBarColorString = expandParams[@"titleBarColor"];
if ([titleBarColorString isKindOfClass:[NSString class]] && [titleBarColorString length] > 0) {
UIColor *titleBarColor = [UIColor colorFromHexString:titleBarColorString];
[self.navigationController.navigationBar setNavigationBarStyleWithColor:titleBarColor translucent:NO];
[self.navigationController.navigationBar setNavigationBarBottomLineColor:titleBarColor];
}

//导航栏是否隐藏，默认不隐藏。设置隐藏后，webview 需全屏
NSString *showTitleBar = expandParams[@"showTitleBar"];
if (showTitleBar && ![showTitleBar boolValue]) {
[self.navigationController setNavigationBarHidden:YES];
}

// 调整 webview 的位置
UIWebView *webView = (UIWebView *)self.psdContentView;
CGRect frame = webView.frame;
frame.origin.y = [[UIApplication sharedApplication] statusBarFrame].size.height;
frame.size.height -= [[UIApplication sharedApplication] statusBarFrame].size.height;
webView.frame = frame;
self.automaticallyAdjustsScrollViewInsets = NO;
}

//导航栏是否透明，默认不透明。设置透明后，webview 需全屏
NSString *transparentTitle = expandParams[@"transparentTitle"];
if ([transparentTitle isEqualToString:@"always"] || [transparentTitle isEqualToString:@"auto"]) {

```

```

// 导航栏和底部横线变为透明
UIColor *clearColor = [UIColor clearColor];
[self.navigationController.navigationBar setNavigationBarTranslucentStyle];
[self.navigationController.navigationBar setNavigationBarStyleWithColor:clearColor translucent:YES];

// 调整 webview 的位置
self.edgesForExtendedLayout = UIRectEdgeAll;
if (@available(iOS 11.0, *)) {
    UIWebView *wb = (UIWebView *)[self psdContentView];
    wb.scrollView.contentInsetAdjustmentBehavior = UIScrollViewContentInsetAdjustmentNever;
}else{
    self.automaticallyAdjustsScrollViewInsets = NO;
}
}

// 修改默认返回按钮文案颜色
NSString *backButtonColorString = expandParams[@"backButtonColor"];
if ([backButtonColorString isKindOfClass:[NSString class]] && [backButtonColorString length] > 0) {
    UIColor *backButtonColor = [UIColor colorFromHexString:backButtonColorString];

    NSArray *leftBarButtonItems = self.navigationItem.leftBarButtonItems;
    if ([leftBarButtonItems count] == 1) {
        if (leftBarButtonItems[0] && [leftBarButtonItems[0] isKindOfClass:[AUBarButton class]]) {
            AUBarButton *backItem = leftBarButtonItems[0];
            backItem.titleColor = backButtonColor;
            backItem.backButtonColor = backButtonColor;
        }
    }
}

// 设置标题颜色
NSString *titleColorString = expandParams[@"titleColor"];
if ([titleColorString isKindOfClass:[NSString class]] && [titleColorString length] > 0) {
    UIColor *titleColor = [UIColor colorFromHexString:titleColorString];
    id <NBNavigationTitleViewProtocol> titleView = self.navigationItem.titleView;
    [[titleView mainTitleLabel] setFont:[UIFont systemFontOfSize:16]];
    [[titleView mainTitleLabel] setTextColor:titleColor];
}
}

```

页面打开后：在用户操作的过程中动态修改导航栏样式。主要通过自定义 JSAPI 的方式，调用 native 接口方法进行修改。

- 自定义 JSAPI 对当前页面导航栏样式进行处理。

参考 定制某一个页面导航栏样式 提供的接口，在 JSAPI 中对原生导航栏进行处理：

```

- (void)handler:(NSDictionary *)data context:(PSDContext *)context
callback:(PSDJsApiResponseCallbackBlock)callback
{
    [super handler:data context:context callback:callback];

    UIViewController *currentVC = context.currentViewController;
    currentVC.navigationItem.titleView = [[AUDoubleTitleView alloc] initWithTitle:@"主标题" detailTitle:@"副标题"];
}

```

```
callback(@{@"success":@YES});  
}
```

3.7.2 H5 容器自动化埋点

使用 mPaaS 提供的 Nebula 容器加载 H5 页面时，Nebula 容器会自动统计页面的行为数据、加载性能，捕获异常报错等数据，方便您更好地跟踪 H5 页面加载时的各项数据。本文将引导您集成 Nebula 容器自动化埋点能力及查看埋点数据。

前置条件

集成 mPaaS 提供的 H5 容器自动化埋点功能，需满足以下两个条件：

- 您已在 [mPaaS 控制台](#) 中创建了一个 App。
- 您已完成 [添加 SDK 步骤](#)，客户端工程集成了 NebulaLogging.framework。

操作步骤

初始化配置

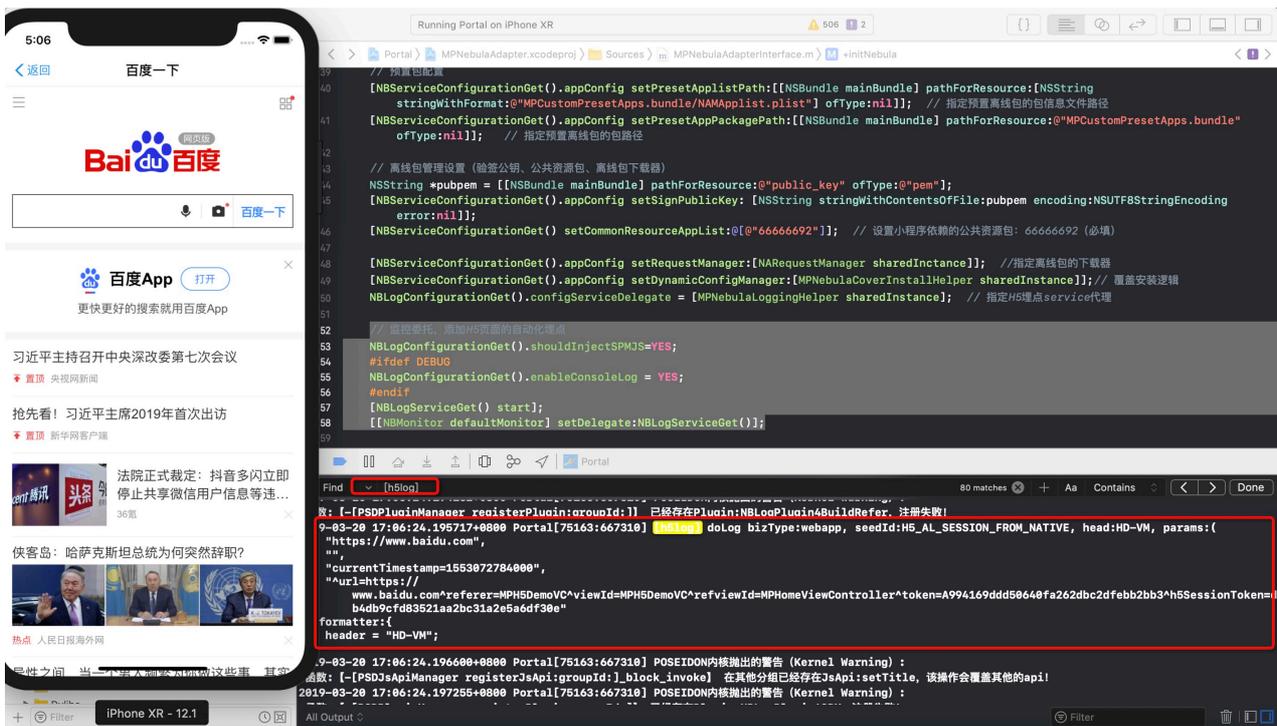
1. 要集成 Nebula 容器的自动埋点功能，需在容器初始化同时启动 H5 埋点监控：

```
// 监控委托，添加 H5 页面的自动化埋点  
NBLogConfigurationGet().shouldInjectSPMJS=YES;  
#ifdef DEBUG  
NBLogConfigurationGet().enableConsoleLog = YES;  
#endif  
[NBLogServiceGet() start];  
[[NBMonitor defaultMonitor] setDelegate:NBLogServiceGet()];
```

2. 启动 H5 埋点监控后，使用 Nebula 容器加载 H5 页面，容器就会自动统计页面加载行为、性能及异常数据等。根据查看埋点数据的场景不同，分为以下两种方式：
 - [查看客户端日志](#)：查看客户端本地的埋点数据，适用于应用开发阶段的问题排查。详见下文 [查看客户端日志](#)。
 - [查看服务端日志](#)：查看线上用户产生的真实埋点数据，适用于应用发布上线后，排查线上问题。详见下文 [查看服务端日志](#)。

查看客户端日志

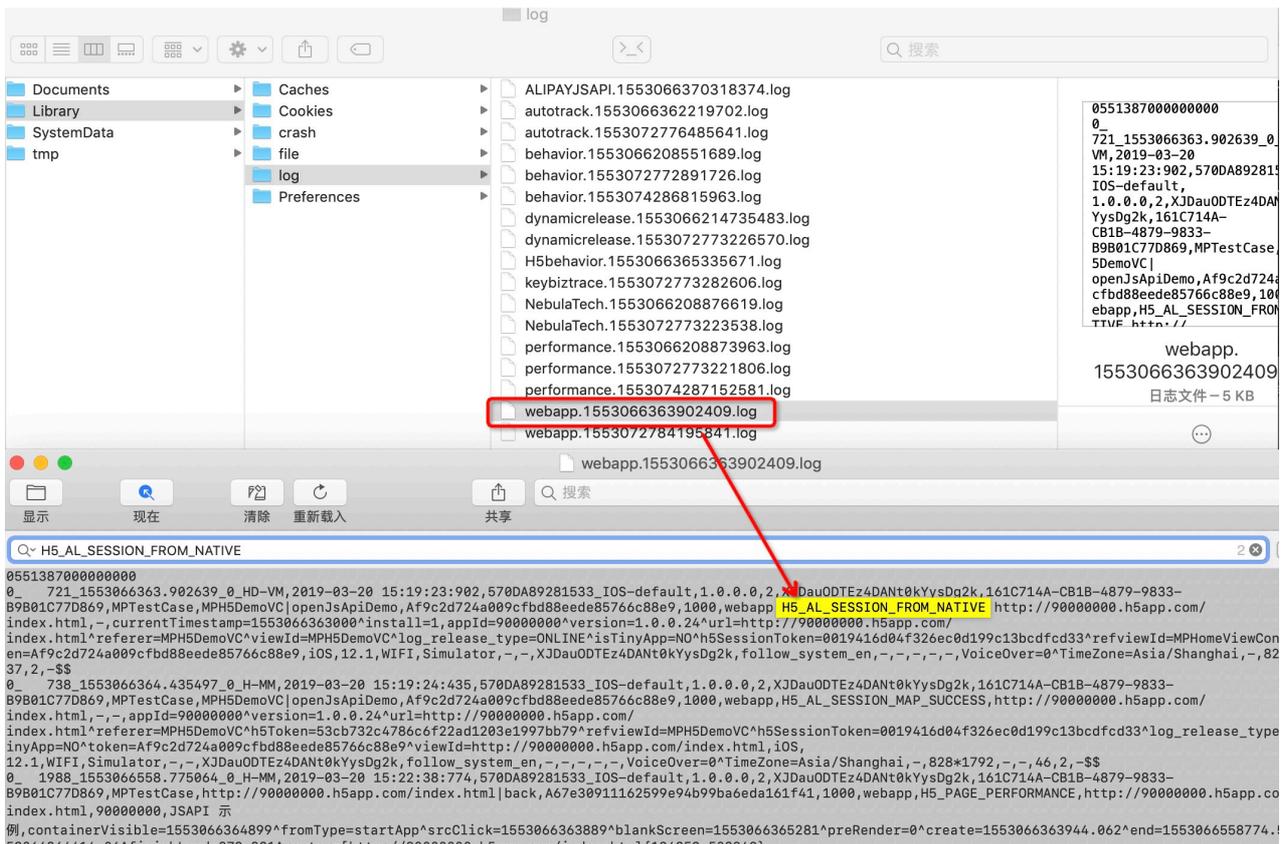
加载 H5 页面后，在 XCode 控制台搜索关键字 [h5log]，即可查看页面加载相关的埋点数据的关键信息，如下图。



Xcode 控制台打出的 H5 容器埋点数据，只包含了以下关键信息：

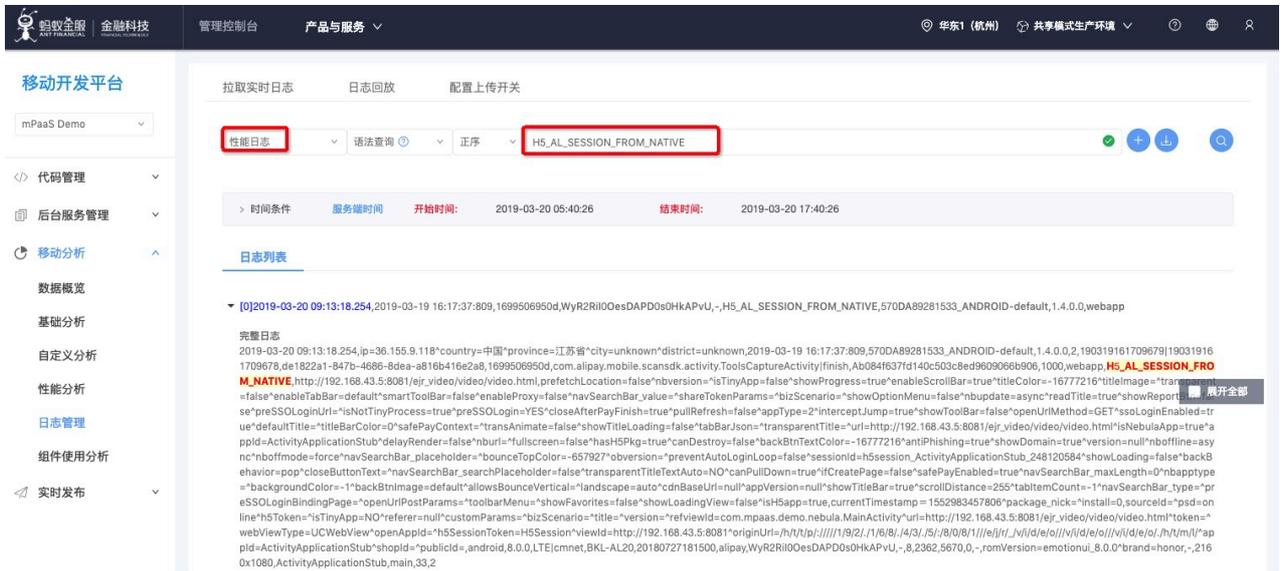
SeedId	示例	含义
bizType	webapp	日志类型，写入本地日志文件的文件名。
seedId	H5_AL_SESSION_FROM_NATIVE	埋点的唯一标识，具体含义参考下文的 H5 容器埋点集。
head	HD-VM	日志模型。可参考 性能埋点。
params	https://www.baidu.com	当前页面的 URL。
	currentTimestamp=1553072784000	自定义参数，如时间戳，页面加载状态。
	-	自定义参数，如页面元素位置。
formatter	header = "HD-VM";	同 head 字段说明。

若您需要查看 Xcode 控制台上某条日志的完整内容，可在沙盒中找到 Library/log 下以 bizType 开头的文件名，然后根据 seedId 关键字搜索查询。



查看服务端日志

若需要查看线上用户 H5 容器的埋点数据，可在 mPaaS 控制台通过 日志回放 进行查询。



H5 容器埋点集

H5 容器统计的自动化埋点信息，根据埋点数据的 seedID 作为唯一标识，按加载的 H5 页面方式不同，主要有以下三类：

- 打开在线 URL 的相关埋点集：

SeedId	含义
H5_AL_SESSION_FROM_NATIVE	容器已经启动。
H5_AL_PAGE_START	页面开始加载。
H5_AL_NETWORK_START	页面开始发起网络请求。
H5_OPEN_PAGE_FINISH	页面加载完成。
H5_AL_PAGE_APPEAR	页面首次出现。
H5_AL_JSAPI_SENDEVENT	页面调用 JsApi。
H5_AL_JSAPI_NOTFOUND	页面调用 JsApi 失败。
H5_TITLEBAR_BACK_BT	点击导航栏返回按钮。
H5_PAGE_PERFORMANCE	统计页面加载的性能。

• 打开离线包页面的相关埋点集：

SeedId	含义
H5_APP_REQUEST	请求离线包信息。
H5_APP_LOAD_DATASOURCE	加载离线包信息。
H5_AL_SESSION_FROM_NATIVE	容器已经启动。
H5_APP_DOWNLOAD	离线包下载。
H5_APP_UNZIP	离线包解压。
H5_APP_POOL	包管理信息池操作（包含池内信息的增、删、改）。
H5_APP_VERIFY	离线包验签。
H5_AL_SESSION_VERIFYTAR_FAIL	离线包验签失败。
H5_AL_PAGE_START	页面开始加载。
H5_AL_SESSION_MAP_SUCCESS	加载本地离线包成功。
H5_AL_SESSION_FALLBACK	加载本地离线包失败，走 fallback 请求在线页面。
H5_OPEN_PAGE_FINISH	页面加载完成。
H5_AL_PAGE_APPEAR	页面首次出现。

• 异常埋点：

seedId	含义
H5_AL_NETWORK_PERFORMANCE_ERROR	资源请求异常。
H5_PAGE_ABNORMAL	页面异常。
H5_AL_PAGE_JSERROR	JS 异常。
H5_AL_JSAPI_RESULT_ERROR	JSAPI 异常。

3.8 iOS 常见问题

- 预置离线包使用H5_json.json文件不生效。
- 如何获取已安装的离线包应用信息？
- 如何强制更新全部离线包信息？
- 在 iOS 13 中，vux-ui pulldown 组件在H5容器中存在滑动卡死的情况。
- 如何查看当前应用已注册的所有 JSAPI 和 Plugins？
- 如何在 JSAPI 或 Plugin 中获取当前 H5 页面的 UIviewController 和 Webview 对象？
- 如何获取当前页面的 WebView？
- 如何获取当前页面加载时前端传入的启动参数？
- 如何限制 JSAPI 只在某个特定离线包中起作用？
- 如何拦截页面 URL？
- 如何在当前 H5 页面加载前，拦截当前页面 url？
- 如何在 Native 手动调用 JSAPI？
- 为什么本地预置的离线包加载不生效？
- 为什么控制台发布新版本离线包后，客户端不能正常加载到新包？
- 全局资源包加载失败？
- 为什么打开某个离线包 H5 页面会白屏或出现 400 错误？
- 怎样禁止H5页面的手势侧滑返回功能？
- 如何判断当前页面是小程序中的页面？
- H5页面如何传递自定义参数？
- 如何H5页面如何传递的参数？
- 如何拦截 JsApi 调用

注意事项：

- H5 容器初始化时，最好指定所有 H5 页面的基类和 WebView 的基类，便于统一处理。

```

70
71
72 + (void)configNebulaSDK
73 {
74     // 控件创建
75     [NBServiceConfigurationGet() setViewControllerClass:[H5WebViewController class]];
76     [NBServiceConfigurationGet() setContentViewControllerClass:[MPWebView class]];
77
78     // 自定义的插件配置，所有自定义的jsapi和plugin都保存在DemoPresetApps.bundle/Poseidon-Extra-Config.plist中
79     NSString *filePath = [[NSBundle mainBundle].bundlePath stringByAppendingFormat:@"%/%/%@",
80         @"DemoPlugins.bundle", @"Poseidon-Extra-Config.plist"];
81     [NBServiceConfigurationGet() setExtraPluginsFilePath:filePath];
82
83     // 预置包配置，所有预置的离线包在DemoPresetApps.bundle中，预置的包信息在 DemoPresetApps.bundle/
84     // NAMApplist.plist 中
85     [NBServiceConfigurationGet().appConfig setPresetApplistPath:[[NSBundle mainBundle] pathForResource:
86         @"NAMApplist.plist" ofType:nil]];
87     [NBServiceConfigurationGet().appConfig setPresetAppPackagePath:[[NSBundle mainBundle]
88         pathForResource:@"DemoPresetApps.bundle" ofType:nil]];
89
90     // 包管理服务配置
91     NSString *version = [[[NSBundle mainBundle] infoDictionary] objectForKey:@"Product Version"];
92     [NBServiceConfigurationGet() setClientVersion:version];
93     NSString *bundleId = [[[NSBundle mainBundle] infoDictionary] objectForKey:@"CFBundleIdentifier"];
94     [NBServiceConfigurationGet() setClientBundleId:bundleId];
95     [NBServiceConfigurationGet() setUserAgent:[NSString stringWithFormat:@"H5Demo %@", version]]; //指定
96     // userAgent
97
98     // 离线包管理设置 (验签公钥、公共资源包、离线包下载器)
99     NSString *pubpem = [[NSBundle mainBundle] pathForResource:@"public_key" ofType:@"pem"];
100    [NBServiceConfigurationGet().appConfig setSignPublicKey: [NSString stringWithContentsOfFile:pubpem
101        encoding:NSUTF8StringEncoding error:nil]]; // 设置离线包验签公钥

```

- 前端在开发离线包时，对应路径及文件名中不允许有中文字符，否则会导致客户端离线包解压失败。
- 离线包中各资源路径的绝对长度不要超过 100，否则导致客户端 tar 包解压失败，页面白屏。

预置离线包使用H5_json.json文件不生效。

解答：在 10.1.32 基线中，只支持 plist 格式。在 10.1.60 基线中，plist 和 json 格式都支持。

如何获取已安装的离线包应用信息？

解答：参考代码 NSDictionary *installedApps = [NAMServiceGet() installApps:nil];。

如何强制更新全部离线包信息？

解答：可以使用封装的 requestAllNebulaApps 方法进行全量更新。

```

[[MPNebulaAdapterInterface sharedInstance]requestAllNebulaApps:^(NSDictionary *data, NSError *error) {
}];

```

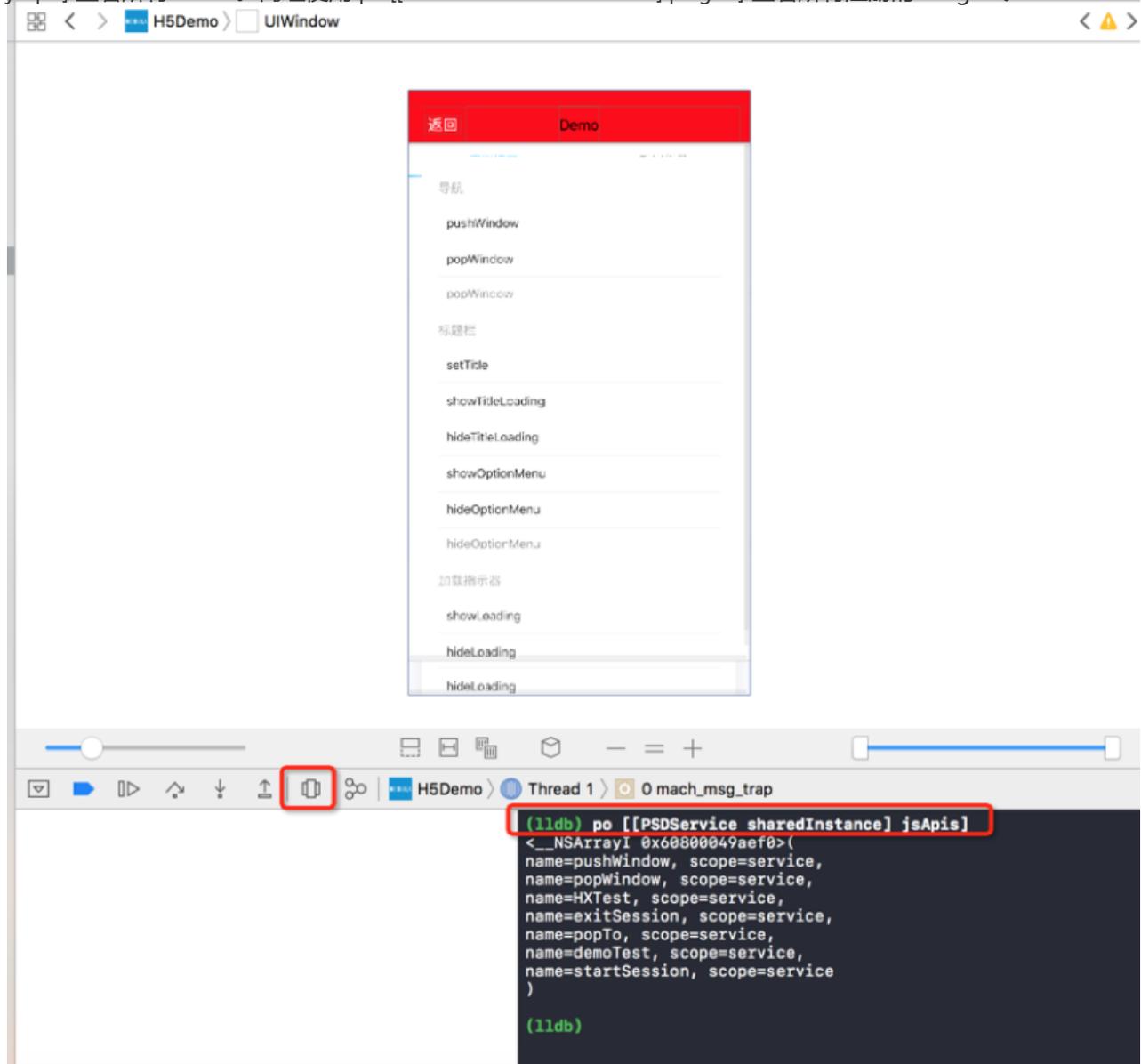
在 iOS 13 中，vux-ui pulldown 组件在H5容器中存在滑动卡死的情况。

解答：这是因为触发了 UIWebView 在 iOS 系统中的 bug，可以考虑切换到 WKWebView 或更换前端组件。切

换为 WKWebView 的方法请参考文档 mPaaS 10.1.60 适配 WKWebView。

如何查看当前应用已注册的所有 JSAPI 和 Plugins ?

解答：打开一个 H5 页面，进入 Xcode View 层级页面，在 lldb 控制台使用 `po [[PSDService sharedInstance] jsApis]` 查看所有 JSAPI。同理使用 `po [[PSDService sharedInstance] plugins]` 查看所有注册的 Plugins。



如何在 JSAPI 或 Plugin 中获取当前 H5 页面的 UIViewController 和 Webview 对象？

解答：在实际执行过程中，Plugin 可直接拿到参数 `event.context`，JSAPI 中可获取到 `context`，在上下文 `PSDContext` 对象中，可以取到您想要的所有信息或引用，比如当前控制器 `event.currentViewController` 的引用，当前 `WebView` 的引用 `context.currentViewController.psdContentView` 等等。

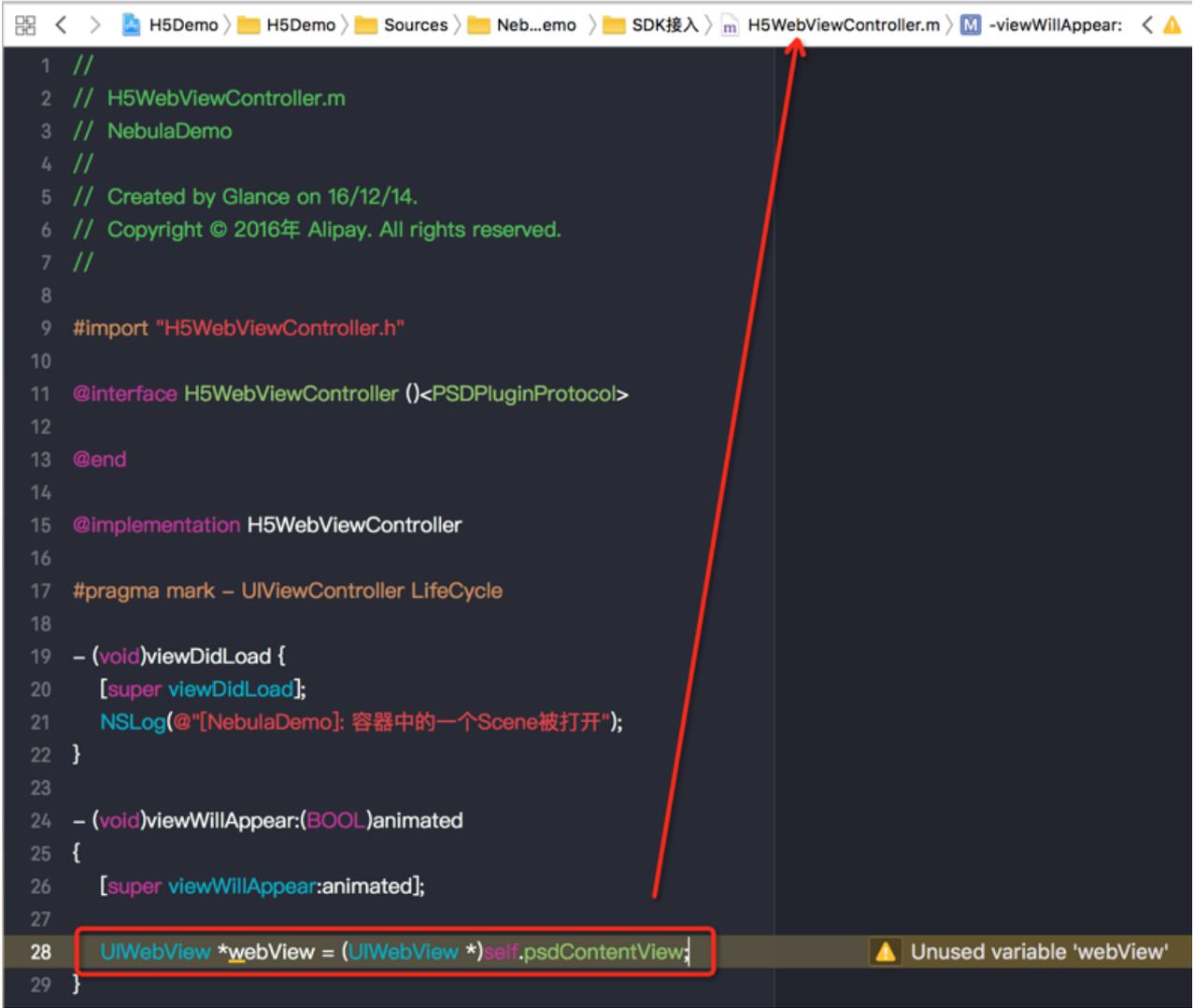
```
H5Demo > H5Demo > Sources > NebulaDemo > 自定义扩展插件 > H5JsApi4DemoTest.m > M -handler:context:callback:
2 // H5JsApi4DemoTest.m
3 // NebulaDemo
4 //
5 // Created by Glance on 16/12/20.
6 // Copyright © 2016年 Alipay. All rights reserved.
7 //
8
9 #import "H5JsApi4DemoTest.h"
10 // 该JSApi已在Poseidon-Extra-Config.plist中注册
11
12 @implementation H5JsApi4DemoTest
13
14 - (void)handler:(NSDictionary *)data
15     context:(PSDContext *)context
16     callback:(PSDJsApiResponseCallbackBlock)callback
17 {
18     [super handler:data context:context callback:callback];
19     UIViewController *vc = context.currentViewController;
20     UIWebView *webView = (UIWebView *)context.currentViewController.psdContentView;
21     NSLog(@"[NEBULADEMO]:JSAPI调用, 当前viewController %@, webView:%@, jsApi实例为 %@",vc, webView, self);
22
23     if (!data) {
24         errorCallback(callback, e_invalid_params);
25         return;
26     }
27     else {
28         callback(@"result":@"jsapi-demoTest来自Native的处理结果");
29     }
30 }
31 }
```

```
H5Demo > H5Demo > Sources > NebulaDemo > 自定义扩展插件 > H5Plugin4DemoTest.m > M -handleEvent:
19         withListener:self
20         useCapture:NO];
21     [super pluginDidLoad];
22 }
23
24 - (void)addJSApis
25 {
26     [super addJSApis];
27     // 这种jsapi的注册是在plist中配置之外的另一种方式
28     PSDJsApi *jsApi4DemoTest2 = [PSDJsApi jsApi:@"demoTest2"
29                                     handler:^(NSDictionary *data, PSDContext *context, PSDJsApiResponseCallbackBlock responseCall
30                                     {
31                                         responseCallbackBlock(@"result":@"jsapi-demoTest2调用Native的处理结果");
32                                     }
33                                     checkParams:NO
34                                     isPrivate:NO
35                                     scope:self.scope];
36     [self registerJsApi2Target:jsApi4DemoTest2];
37 }
38 - (void)handleEvent:(PSDEvent *)event
39 {
40     [super handleEvent:event];
41     UIViewController *vc = event.context.currentViewController;
42     UIWebView *webView = (UIWebView *)event.context.currentViewController.psdContentView;
43     NSString *eventType = event.eventType;
44     NSLog(@"[NEBULADEMO]:有事件抛出, 当前viewController %@, webView:%@, 事件名为 %@",vc, webView, eventType);
45 }
46 }
47
48
```

如何获取当前页面的 WebView?

解答：当前 H5 页面的 WebView 是当前 VC 的一个属性，可通过 vc.psdContentView 获取，在 JSAPI 或 Plugin 中可通过上面的方法获取当前页面的 VC。

注意：在 H5 页面基类获取当前页面的 WebView 时，请在 viewWillAppear 方法中。viewDidLoad 方法中 WebView 未创建，如果使用该方法，取到的值会为 nil。

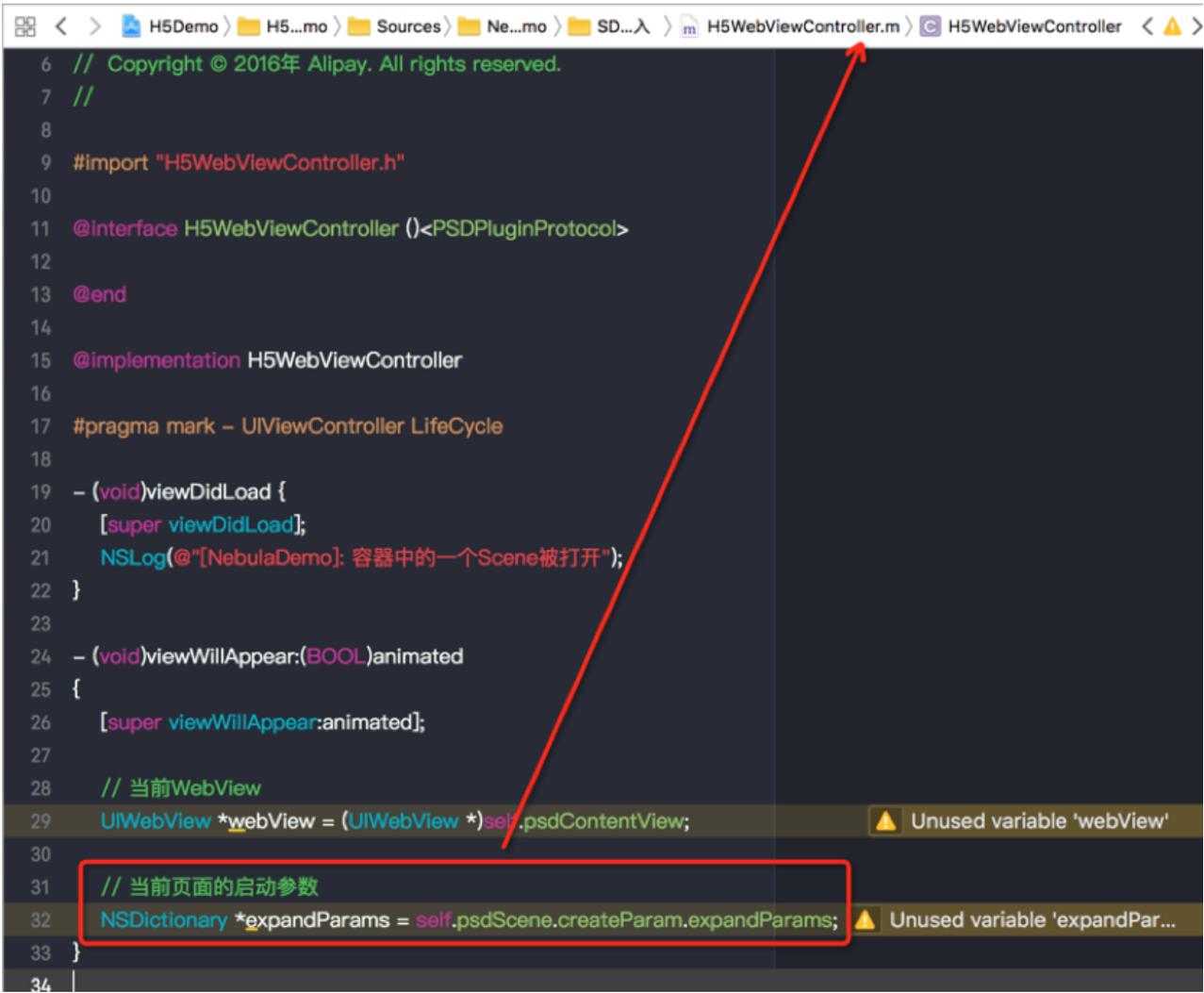


```
1 //
2 // H5WebViewController.m
3 // NebulaDemo
4 //
5 // Created by Glance on 16/12/14.
6 // Copyright © 2016年 Alipay. All rights reserved.
7 //
8
9 #import "H5WebViewController.h"
10
11 @interface H5WebViewController ()<PSDPluginProtocol>
12
13 @end
14
15 @implementation H5WebViewController
16
17 #pragma mark - UIViewController LifeCycle
18
19 - (void)viewDidLoad {
20     [super viewDidLoad];
21     NSLog(@"[NebulaDemo]: 容器中的一个Scene被打开");
22 }
23
24 - (void)viewWillAppear:(BOOL)animated
25 {
26     [super viewWillAppear:animated];
27
28     UIWebView *webView = (UIWebView *)self.psdContentView;
29 }
```

Unused variable 'webView'

如何获取当前页面加载时前端传入的启动参数？

解答：直接获取当前 VC 的 `psdScene.createParam.expandParams` 属性。



```
6 // Copyright © 2016年 Alipay. All rights reserved.
7 //
8
9 #import "H5WebViewController.h"
10
11 @interface H5WebViewController ()<PSDPluginProtocol>
12
13 @end
14
15 @implementation H5WebViewController
16
17 #pragma mark - UIViewController LifeCycle
18
19 - (void)viewDidLoad {
20     [super viewDidLoad];
21     NSLog(@"[NebulaDemo]: 容器中的一个Scene被打开");
22 }
23
24 - (void)viewWillAppear:(BOOL)animated
25 {
26     [super viewWillAppear:animated];
27
28     // 当前WebView
29     UIWebView *webView = (UIWebView *)self.psdContentView;
30
31     // 当前页面的启动参数
32     NSDictionary *expandParams = self.psdScene.createParam.expandParams;
33 }
34
```

如何限制 JSAPI 只在某个特定离线包中起作用？

解答：在 JSAPI 的实际执行方法中，获取当前页面所属离线包的 appId，决定是否执行逻辑就可以了。

```

H5Demo > H5...mo > So...es > Ne...mo > 自...件 > H5JsApi4DemoTest.m > M -handler:context:callback: < >
2 // H5JsApi4DemoTest.m
3 // NebulaDemo
4 //
5 // Created by Glance on 16/12/20.
6 // Copyright © 2016年 Alipay. All rights reserved.
7 //
8
9 #import "H5JsApi4DemoTest.h"
10 // 该JSApi已在Poseidon-Extra-Config.plist中注册
11
12 @implementation H5JsApi4DemoTest
13
14 - (void)handler:(NSDictionary *)data
15     context:(PSDContext *)context
16     callback:(PSDJsApiResponseCallbackBlock)callback
17 {
18     [super handler:data context:context callback:callback];
19     UIViewController *vc = context.currentViewController;
20     UIWebView *webView = (UIWebView *)context.currentViewController.psdContentView;
21     NSLog(@"[NEBULADEMO]:JSAPI调用, 当前viewController %@, webView:%@, jsApi实例为 %@",vc, webView,
22         self);
23     NSString *appId = context.currentSession.createParam.expandParams[@"appId"];
24
25     if (!data) {
26         errorCallback(callback, e_invalid_params);
27         return;
28     }
29     else {
30         callback(@"result":@"jsapi-demoTest来自Native的处理结果");
31     }
32 }
33 }

```

如何拦截页面 URL ?

解答：您可以自定义 Plugin，通过监听事件来实现。

- 监听事件名称[PSDProxy addEventListener:kEvent_Proxy_Request_Start_Handler withListener:self useCapture:YES];。
- 拦截处理

```

else if ([kEvent_Proxy_Request_Start_Handler isEqualToString:event.eventType]
&& [event isKindOfClass:[PSDProxyEvent class]]) {

NSLog(@"-----kNBEvent_Scene_NavigationItem_Right_Setting_Click----");
PSDProxyEvent *proxyEvent = (PSDProxyEvent*) event;
NSMutableURLRequest *redirectReq = proxyEvent.request.mutableCopy;
NSString *appId = event.context.currentSession.createParam.expandParams[@"appId"];

NAMApp *app = [NAMServiceGet() findApp:appId version:nil];

```

```

NSString *fallBackUrl = app.fallback_host;
NSString *vhost = app.vhost;;
NSString *url = redirectReq.URL.absoluteString;

NSLog(@"url_%@_ fallBackUrl:%@",url,fallBackUrl);
if ([url containsString:@"www.baidu.com"]) {
    [proxyEvent preventDefault];
}
}

```

如何在当前 H5 页面加载前，拦截当前页面 url ？

解答：在 H5 页面的基类中，实现 UIWebView 的生命周期的代理方法中，监听 kEvent_Navigation_Start 事件，在页面加载之前拦截，获取当前页面的 WebView 和 URL 进行相关处理。

```

73 #pragma mark - 对应UIWebViewDelegate的委托实现
74
75 - (void)handleEvent:(PSDEvent *)event
76 {
77     if (![event.context currentViewController] isEqual:self) {
78         return;
79     }
80     if ([kEvent_Navigation_Start isEqualToString:event.eventType]) {
81         BOOL shouldStart = [self handleContentViewShouldStartLoad:(id)event];
82
83         if (!shouldStart) {
84             [event preventDefault];
85         }
86     }
87     else if ([kEvent_Page_Load_Start isEqualToString:event.eventType]) {
88         [self handleContentViewDidStartLoad:(id)event];
89     }
90     else if ([kEvent_Page_Load_Complete isEqualToString:event.eventType]) {
91         [self handleContentViewDidFinishLoad:(id)event];
92     }
93     else if ([kEvent_Navigation_Error isEqualToString:event.eventType]) {
94         [self handleContentViewDidFailLoad:(id)event];
95     }
96     else if ([kNBEvent_Scene_NavigationItem_Left_Back_Click isEqualToString:event.eventType]) {
97
98     }
99 }
100
101
102 - (BOOL)handleContentViewShouldStartLoad:(PSDNavigationEvent *)event
103 {
104     return YES;
105 }

```

如何在 Native 手动调用 JSAPI ？

解答：有时 Native 端可能需要您在当前页面手动调用某个 JSAPI 接口，可通过调用当前 VC 的以下接口实现

```

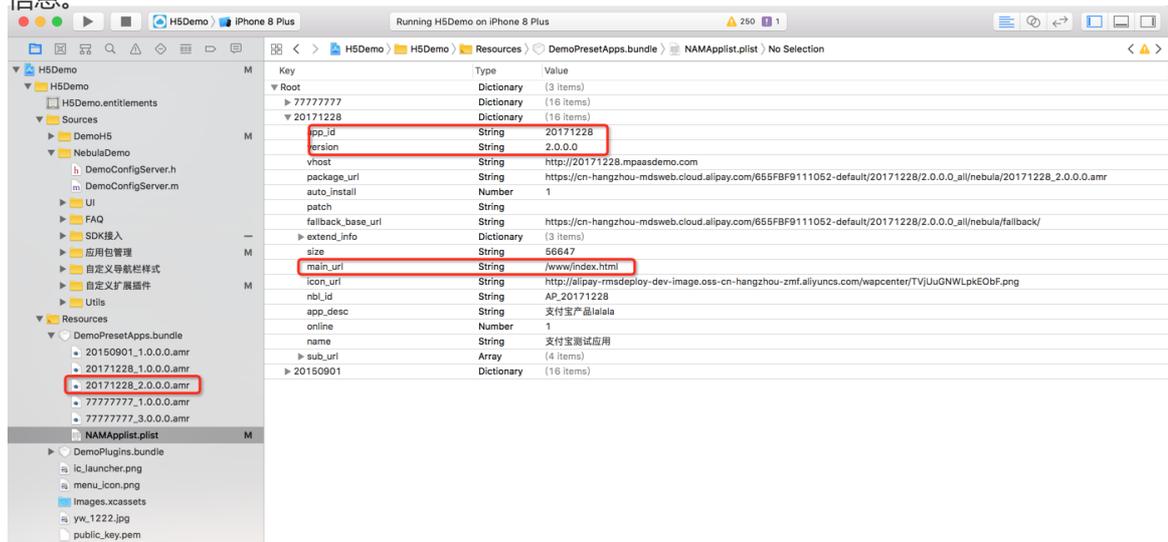
50 if (self.navigationController) {
51     NSMutableArray *arr = [[self.navigationController viewControllers] mutableCopy];
52     [arr removeObject];
53
54     [self.navigationController setViewControllers:arr animated:YES];
55 }
56 else {
57     [self dismissViewControllerAnimated:YES completion:NULL];
58 }
59 }
60
61 - (IBAction)btnRightItemClicked:(id)sender
62 {
63     NSLog(@"[NebulaDemo]: 导航栏右侧按钮被点击");
64
65     // native手动触发 JSAPI
66     [self callHandler:@"HXTest" data:@{@"key":@"value"} responseCallback:^(id responseData) {
67
68     }];
69 }
70

```

为什么本地预置的离线包加载不生效？

解答：预置资源包加载失败一般为预置包版本和包信息不匹配，测试本地预置离线包时，请先断开网络，避免离线包有更新，确保加载的是客户端本地预置的版本。

- 检查本地预置的离线包信息与 Plist 中配置的包信息是否一致，包括 appID、version、main_url 等信息。

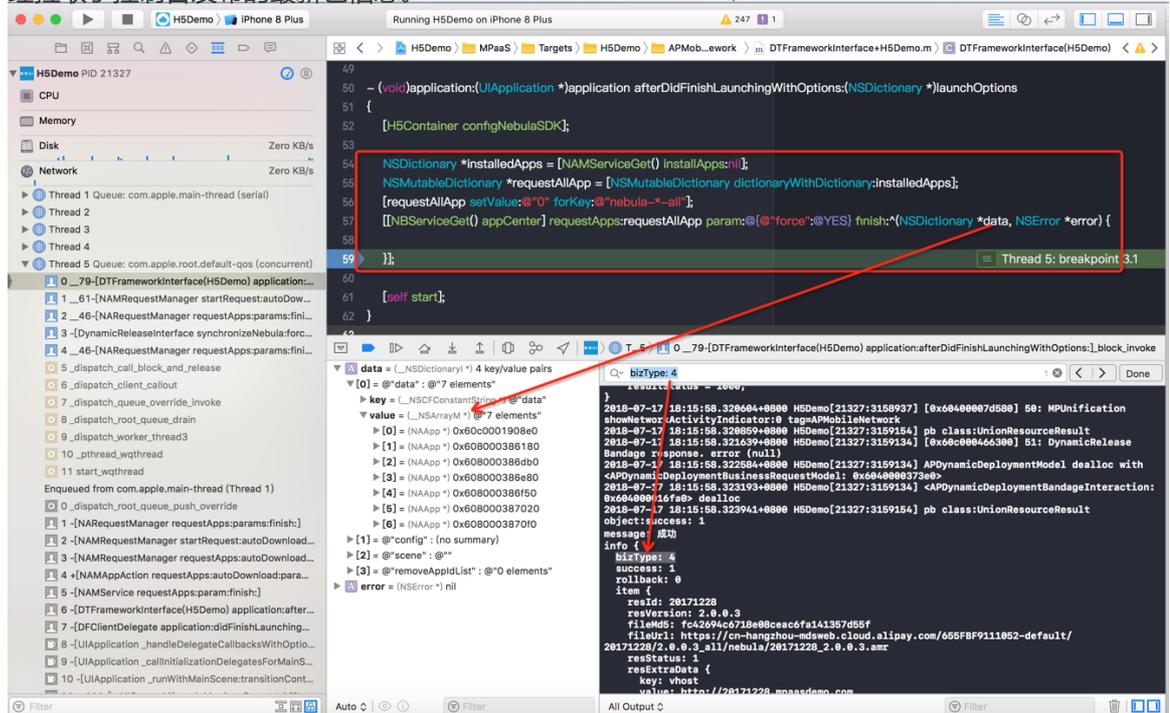


为什么控制台发布新版本离线包后，客户端不能正常加载到新包？

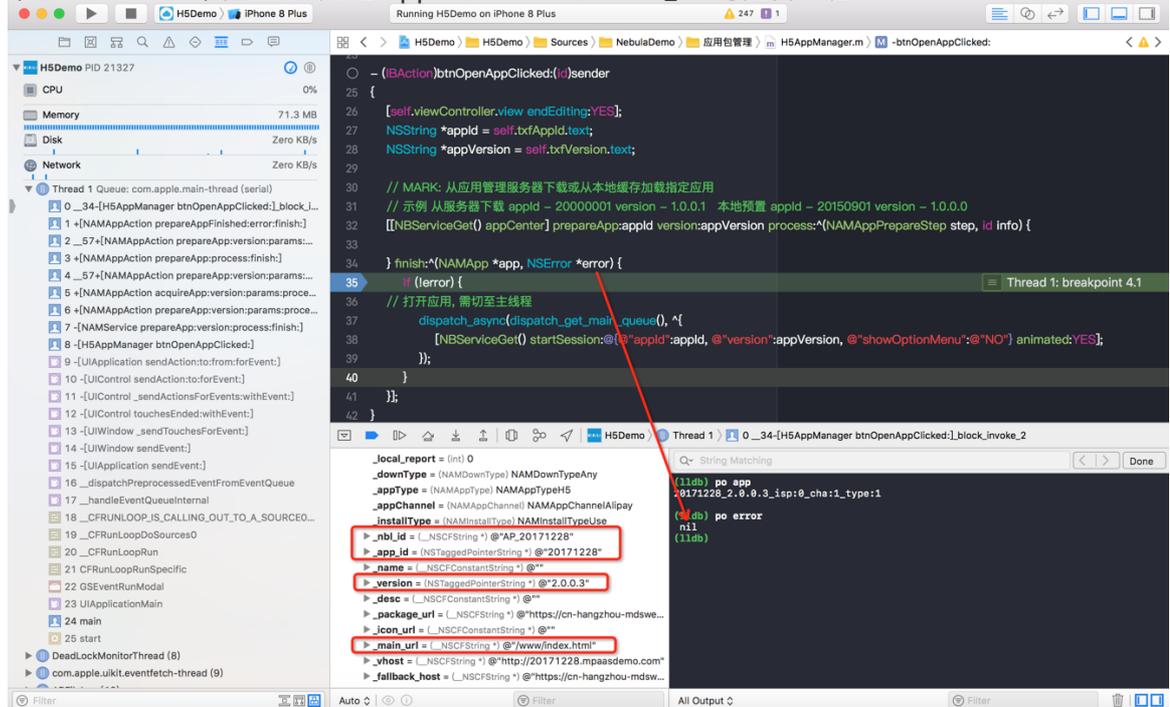
解答：在查看此问题解决方案前，确认您已理解 离线包更新原理，客户端不能正常加载新包，离线包渲染在任

一阶段都有可能出错，下面将一一进行排查：

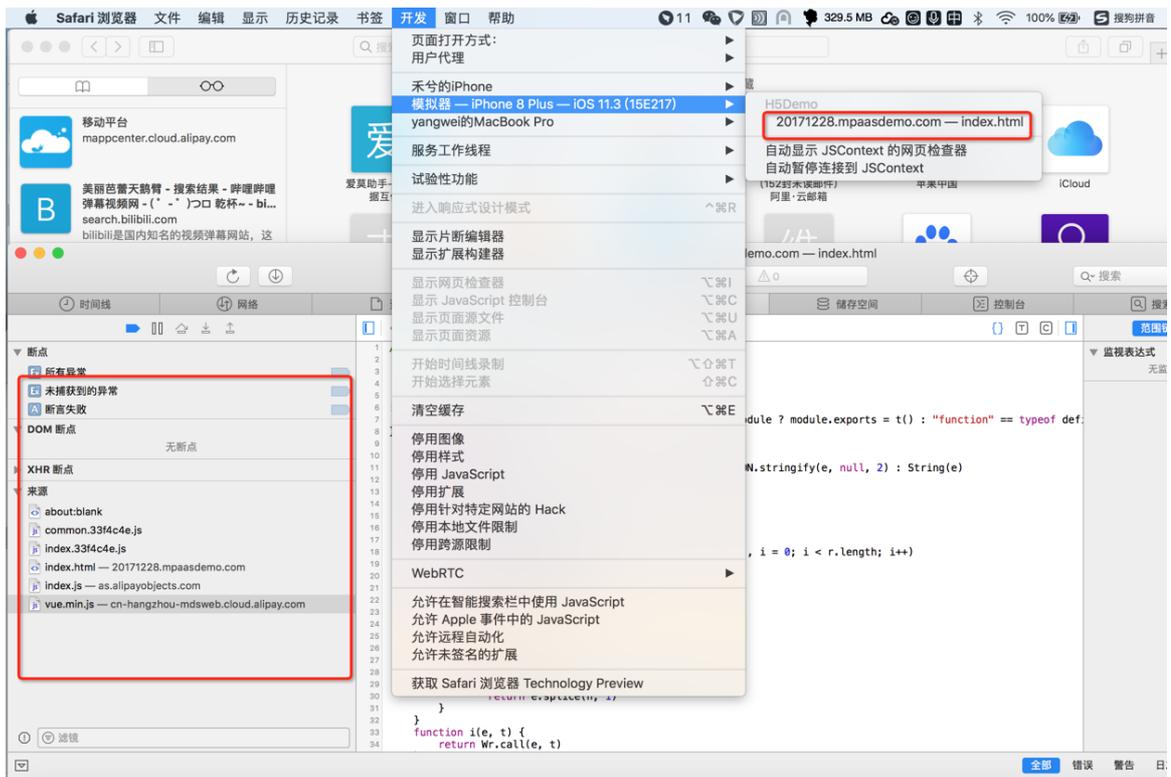
查看全量更新离线包的 RPC 返回结果，在控制台搜索 “bizType: 4” 确认返回的离线包详情，确认已经拉取了控制台发布的最新包信息。



上一步检查通过后，在加载离线包的 finish 回调方法里查看离线包信息，确保为控制台发布的最新包，且 error = nil，检查离线包的 appID、version、main_url 等是否正确。



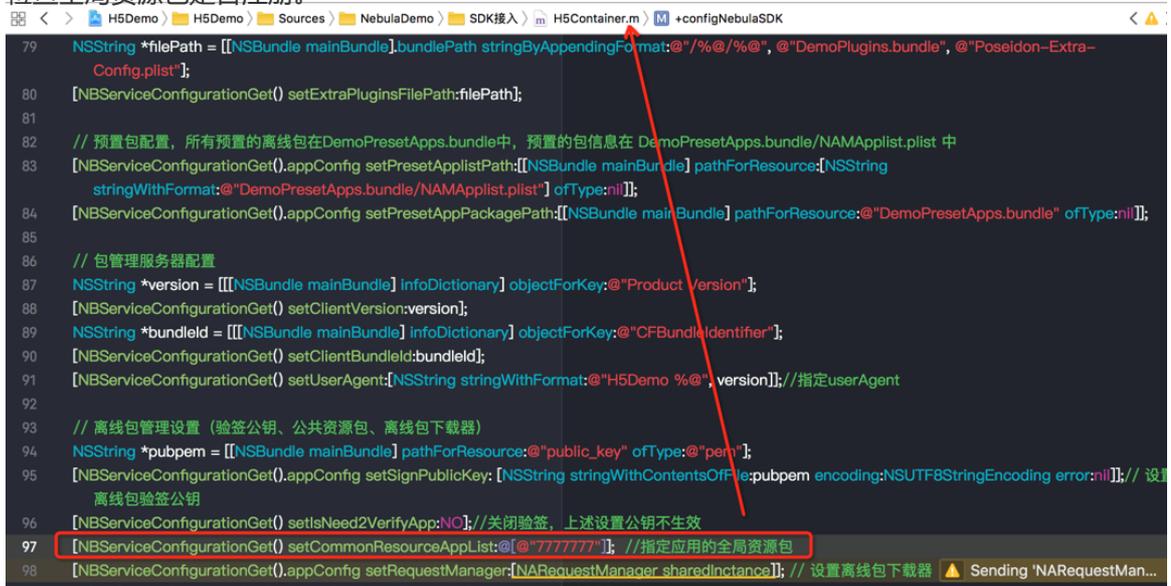
上一步检查通过后，查看沙盒目录下离线包是否解压成功（若当前离线包有引用全局资源包中内容，此解压目录下同样需要有全局资源包）。



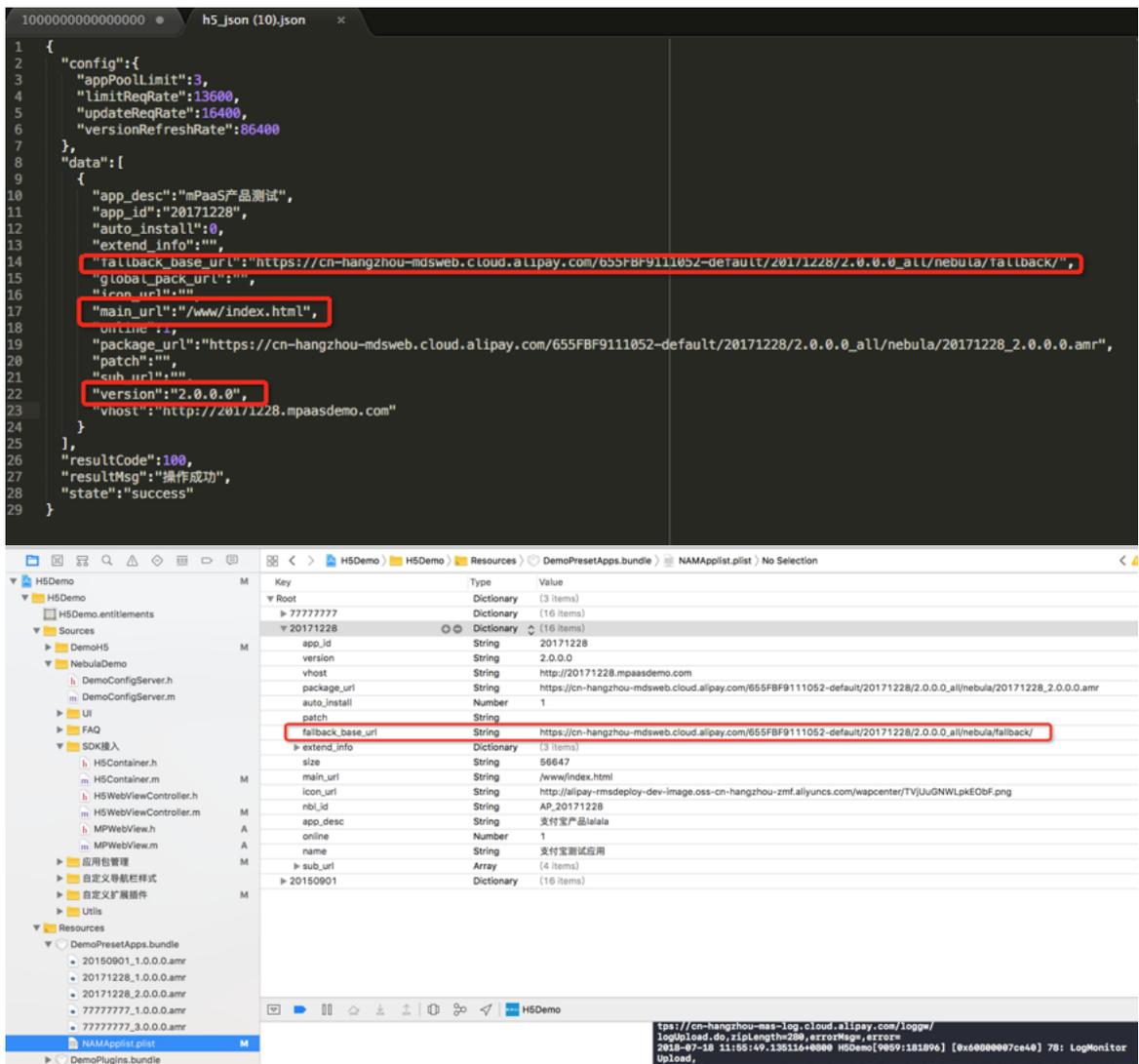
全局资源包加载失败？

解答：当通过 Safari 调试判断出全局资源包加载失败时，可按下述步骤具体排查：

检查全局资源包是否注册。



上一步检查通过后，查看沙盒目录下离线包是否解压成功。



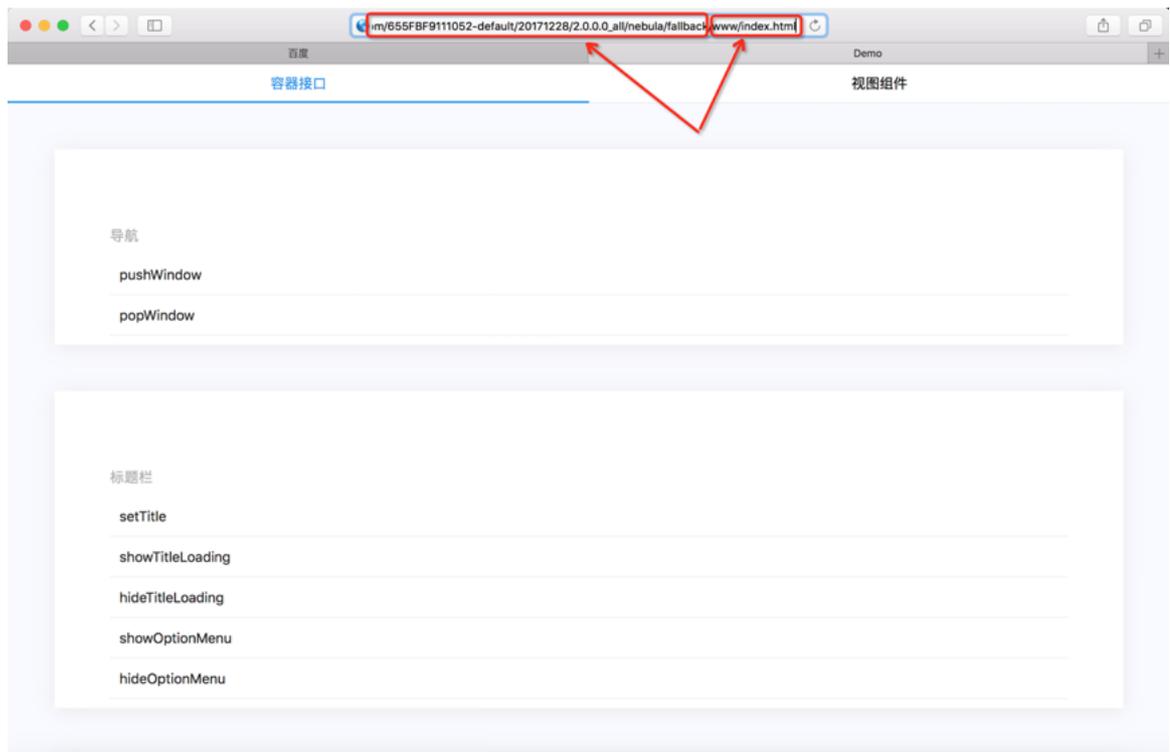
The screenshot displays a code editor with a JSON configuration file named 'h5_json (10).json'. The configuration includes a 'config' object with settings like 'appPoolLimit', 'limitReqRate', 'updateReqRate', and 'versionRefreshRate'. The 'data' array contains an object for an application with fields such as 'app_desc', 'app_id', 'auto_install', 'global_pack_url', 'icon_url', 'main_url', 'package_url', 'patch', 'sub_url', 'version', and 'vhost'. Several fields are highlighted with red boxes: 'fallback_base_url', 'main_url', 'version', and 'fallback_base_url' in the dictionary view.

Below the code editor, a resource dictionary is shown for the application '20171228'. The dictionary lists various keys and their values, including 'app_id', 'version', 'vhost', 'package_url', 'auto_install', 'patch', 'fallback_base_url', 'extend_info', 'size', 'main_url', 'icon_url', 'nbi_id', 'app_desc', 'online', 'name', and 'sub_url'. The 'fallback_base_url' value is highlighted with a red box.

At the bottom right, a log window shows the following message:

```
tps://cn-hangzhou-mas-log.cloud.alipay.com/loggw/
logUpload.do;zipLength=209;errorCode=error=
2018-07-18 11:05:49.136116+0000 H5Demo[9869:181896] [0x60000007ce40] 78: LogMonitor
Upload,
```

- 并且 fallback_base_url + main_url拼接的地址在浏览器可正常加载。



怎样禁止H5页面的手势侧滑返回功能？

解答：支持由前端 H5 页面禁止和 原生 H5 容器基类禁止。

前端 H5 页面禁止：调用 setGestureBack JSAPI 实现。适用于某一个页面需要禁止手势侧滑返回的场景 AlipayJSBridge.call('setGestureBack',{val:false});。

原生 H5 容器基类禁止：在基类的 viewDidAppear 方法中调用系统禁止侧滑返回的接口，同时设置 guestBack 参数。适用于多个或所有 H5 页面需要禁止手势侧滑返回的场景。

```
- (void)viewDidAppear:(BOOL)animated {
    [super viewDidAppear:animated];

    self.options.gestureBack = NO;
    if ([self.navigationController respondsToSelector:@selector(interactivePopGestureRecognizer)]) {
        self.navigationController.interactivePopGestureRecognizer.enabled = NO;
    }
}
```

如何判断当前页面是小程序中的页面？

解答：获取当前页面所在的 session，调用 isTinyAppWithSession 接口判断。代码示例如下：

```
PSDSession *session = self.psdSession;
BOOL isTinyApp = [NBUtils isTinyAppWithSession:session];
```

H5页面如何传递自定义参数？

解答：根据传参方式，分为一下几种场景：

原生 - H5：调用 startH5ViewControllerWithParams 方法时传递[[MPNebulaAdapterInterface sharedInstance] startH5ViewControllerWithParams:@{@ "url": @"https://tech.antfin.com", @ "key1":@"value1"}];。

原生 - 离线包：调用 startH5ViewControllerWithNebulaApp 方法时传递[[MPNebulaAdapterInterface sharedInstance] startH5ViewControllerWithNebulaApp:@{@ "appId":@"70000000",@ "param":@{@ "key2":@"value2"}];。

H5 - H5：调用 pushWindow 时将自定义参数写在 passData 中

```
AlipayJSBridge.call('pushWindow', {
// 要打开页面的 URL
url: 'https://m.taobao.com/',
// 打开页面的配置参数
param: {
readTitle: true, //自动读取 title
showOptionsMenu: false, // 隐藏右边菜单
transparentTitle:'always',
},
// 用于给新开的页面传递参数，可选
// 在新开的页面使用 AlipayJSBridge.startupParams 可以获取到 passData
passData: {
key1:"key1Value",
key2:"key2Value"
}
});
```

H5 - 离线包：调用 startApp jsapi 时将自定义参数写在 param 中

```
AlipayJSBridge.call('startApp', {
appId: '70000000',
param: {
key1:'value1'
}
}, function(result) {
// noop
});
```

如何H5页面如何传递的参数？

解答：分为前端获取和原生获取两种场景：

前端获取：通过 startupParams 方法获取。

```
// 当前页面的启动参数
AlipayJSBridge.startupParams
```

原生获取：通过当前页面所在的 VC 对象获取。

```
// 当前页面的启动参数
NSDictionary *expandParams = self.psdScene.createParam.expandParams;
NSLog(@"[mpaas] expandParams: %@", expandParams);
```

如何拦截 JsApi 调用

解答：您可以自定义 Plugin，通过监听事件来实现。

- 监听事件名称：kEvent_Invocation_Event_Start。
- 拦截处理：获取到 JsApi 的名称、传递的参数等。

```
else if([kEvent_Invocation_Event_Start isEqualToString:event.eventType]) {
    PSDInvocationEvent * invocationEvent = (PSDInvocationEvent *)event;
    NSString * apiName = invocationEvent.invocationName;
    if([apiName isEqualToString:@"setOptionsMenu"] || [apiName isEqualToString:@"showOptionsMenu"]) {
        NSLog(@"www");
    }
}
```

4 内置 JSAPI

4.1 启动参数

H5 容器运行时的外观和行为受一组参数控制，可在启动一个新实例或者 pushWindow 时指定，例如：

```
alipays://platformapi/startapp?appId=20000067&url=http%3A%2F%2Fm.taobao.com&showOptionsMenu=NO&startMultApp=YES
```

另外，新实例的第一个 URL 也可以带一个魔法参数 `webview_options`，其内容将被容器取出并传给容器本身：

```
?webview_options=showOptionsMenu%3DNO&startMultApp%3DYES
// urlencode('showOptionsMenu=NO&startMultApp=YES') => showOptionsMenu%3DNO&startMultApp%3DYES
```

客户端透传给前端的启动参数，前端可以直接通过 `AlipayJSBridge.startupParams` 或 `jsapi:getStartupParams` 获取。

名称	说明	默认值	pushWindow 可

				用	
url			起始 URL	""	Y
defaultTitle			默认标题，在页面第一次加载之前显示在标题栏上	""	Y
showLoading			YES/NO，是否在页面加载前显示全局菊花	"NO"	Y
readTitle			YES/NO，是否读取网页标题显示在 titleBar 上	"YES"	Y
businessScenario			业务场景来源，这个值会记录到每一个埋点中，可以用来区分不同来源。	""	-
backBehavior			back/pop/auto 指定后退按钮行为。 back : 如存在浏览器历史则后退上一页，否则关闭当前 webview。 pop : 直接关闭当前窗口。 auto : 在 iOS 上相当于 pop；在 Android 上，toolbar 可见时相当于 back，toolbar 不可见时相当于 pop。	非 H5App 的通用浏览器模式 (app Id 为 20000067) 为 back，H5 App (用 startActivity 来启动) 为 pop	-
pullRefresh			YES/NO，是否支持下拉刷新。 只有本地文件允许设置为 YES	"NO"	Y
toolbarMenu			JSON 字符串，更多的菜单项列表（放在分享、字号、复制链接后面） 例：{ "menus" :[{ "name" : " 恭喜", " icon" : " H5Service.bundle/h5_popovermenu_share", " action" : " hello" }, { "name" : " 发财", " icon" : " H5Service.bundle/h5_popovermenu_abuse", " action" : " world" }]}	""	Y
showProgress			YES/NO，是否显示加载的进度条	"NO"	-
canPullDown			YES/NO，页面是否支持下拉（显示出黑色背景或者域名） 只有本地文件允许设置为 NO	"YES"	YES
showDomain			YES/NO，页面下拉时是否显示域名 只有本地文件允许设置为 NO，离线包强制设置为 NO，不容许显示	"YES"	-

backgr ound Color		设置背景颜色（十进制，例如：bc=16775138）	""	-	
showO ption Menu		YES/NO，是否显示右上角的“...”按钮	对于 H5Ap p 为 NO 对于 非 H5Ap p 为 YES		
showTi tleL oadin g		YES/NO，是否在 TitleBar 的标题左边显示小菊花）	NO	Y	
ena ble Scr oll Bar		YES/NO，是否使用 webview 的滚动条，包括垂直和水平。只对 Android 有效	默认 为 “YES ”	-	

4.2 事件扩展

4.2.1 初始化完毕

当 window.onload 后，容器会初始化，产生全局变量 AlipayJSBridge，然后触发 JS Bridge 初始化完毕（AlipayJSBridgeReady）事件。

AlipayJSBridge 注入是一个异步过程，因此尽可能监听该事件后，再调用 AlipayJSBridgeReady 接口。

AlipayJSBridgeReady 接口使用方法

```
function ready(callback) {
  // 如果 jsbridge 已经注入则直接调用
  if (window.AlipayJSBridge) {
    callback && callback();
  } else {
    // 如果没有注入则监听注入的事件
    document.addEventListener('AlipayJSBridgeReady', callback, false);
  }
}
```

代码示例

以下代码示例为 bridge 入口的标准写法：

```
<h1>bridge 使用方法</h1>
```

```
<script>
function ready(callback) {
if (window.AlipayJSBridge) {
callback && callback();
} else {
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}

ready(function() {
alert('bridge ready');
});
</script>
```

4.2.2 点击标题栏

点击导航栏标题（titleClick）触发回调。

titleClick 接口使用方法

```
document.addEventListener('titleClick', function(e) {
alert('title clicked')
}, false);
```

代码示例

以下代码示例为基本功能演示：

```
<h1>请点击标题查看效果</h1>

<script>
document.addEventListener('titleClick', function(e) {
alert('title clicked')
}, false);
</script>
```

4.2.3 点击副标题栏

点击导航栏副标题（subtitleClick）触发回调。

subtitleClick 接口使用方法

```
document.addEventListener('subtitleClick', function (e) {
alert('subtitle clicked')
}, false);
```

代码示例

以下代码示例为基本功能演示：

```
<h1>请点击副标题查看效果</h1>
<script>
document.addEventListener('subtitleClick', function(e) {
alert('subtitle clicked')
}, false);

function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}
ready(function() {
AlipayJSBridge.call('setTitle', {
title: '标题',
subtitle: '副标题'
});
});
</script>
```

4.2.4 页面压后台

当一个 webview 界面不可见时，例如被压入后台、锁屏、或 pushwindow 到下个页面，会触发页面压后台（pause）事件。

注意：

- 对于 10.0.15 以上版本的客户端，容器新增了 pagePause 和 appPause 事件，用于业务区分是哪一种情况触发 pause。
- 页面压后台（pause）= 客户端压后台不可见（appPause）+ 窗口压栈底不可见（pagePause）。
- 由于 Android 系统无法区分原生的 resume 和 pause 事件是由于压后台导致的还是由于页面切换导致的，所以 pageResume 和 pagePause 事件是通过 JSAPI 调用记录去回调的，所以仅适用于同一个 session 内 window 之间的互相切换。对于 startApp 和其他客户端直接切换页面方式不生效，例如 chooseImage。

pause 接口使用方法

```
document.addEventListener('pause', function(e) {
alert("pause");
}, false);
```

代码示例

以下示例为离开当前页面后弹出警告：

```
<h1>请点击按钮打开一个新窗口</h1>
<a href="javascript:void(0)"class="btn J_new_window">新窗口打开当前页面</a>
<script>
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}
ready(function() {
document.querySelector('.J_new_window').addEventListener('click', function() {
AlipayJSBridge.call('pushWindow', {
url: location.pathname,
});
});
});

document.addEventListener('pause', function(e) {
alert('paused');
}, false);
});
</script>
```

4.2.5 页面恢复运行

当一个 webview 界面重新回到栈顶时，例如从后台被唤起、锁屏界面恢复、从下个页面回退，会触发页面恢复运行（resume）事件。

如果这个界面是通过 popWindow 或 popTo 到达，且传递了 data 参数，则此页可以获取到这些参数。

如果在界面的 resume 之前先发生了 App 的 resume，则 event 还会有一个 resumeParams，包含 app resume 时接收到的参数。

注意：

- 对于 10.0.15 以上版本的客户端，容器新增了 pageResume 和 appResume 事件，用于业务区分是哪种情况触发 resume。
- 页面恢复运行（resume）= 客户端从后台被唤起、锁屏界面恢复（appResume）+ 窗口从下个页面回退后恢复显示（pageResume）。
- 由于 Android 原生的 resume 和 pause 事件不能区分是压后台导致还是页面切换导致，所以 pageResume 和 pagePause 事件是通过 JSAPI 调用记录去回调的，所以仅适用于同一个 session 内 window 之间的互相切换。对于 startApp 和其他客户端直接切换页面方式不生效，例如 chooseImage。

resume 接口使用方法

```
document.addEventListener('resume', function(e) {
  console.log("resumed");
}, false);
```

代码示例

以下示例为返回的时候带有数据：

```
<h1>点击"打开新页面", 然后返回时, 会带数据回这个页面</h1>
<a href="#" class="btn J_demo">打开新页面</a>
<script>
function ready(callback) {
  // 如果 jsbridge 已经注入则直接调用
  if (window.AlipayJSBridge) {
    callback && callback();
  } else {
    // 如果没有注入则监听注入的事件
    document.addEventListener('AlipayJSBridgeReady', callback, false);
  }
}
ready(function() {
  document.querySelector('a').addEventListener('click', function() {
    AlipayJSBridge.call('pushWindow', {
      url: location.pathname
    });
  });
});

document.addEventListener('back', function(e) {
  e.preventDefault();

  AlipayJSBridge.call('popWindow', {
    data: {
      from: location.href,
      info: Date.now()
    }
  });
});

document.addEventListener('resume', function(event) {
  alert('页面回退时带过来的内容: ' + JSON.stringify(event.data));
});
</script>
```

API

出参

事件处理函数的 event 参数有以下属性：

名称	类型	描述
data	object	当前 App 实例内 popTo 或者 popWindow 带过来的内容，无法跨 appId 传递数据
resumeParams	object	包含 app resume 时接收到的参数

4.2.6 点击右上角按钮

当您调用了 `setOptionsMenu` 接口自定义了导航栏右上角按钮以后，点击按钮时触发该事件（`optionMenu`）。

optionMenu 接口使用方法

```
document.addEventListener('optionMenu', function (e) {  
  alert("option menu clicked");  
}, false);
```

代码示例

以下示例为基本功能演示：

```
<h1>请点击右上角菜单查看效果</h1>  
<script>  
function ready(callback) {  
  // 如果 jsbridge 已经注入则直接调用  
  if (window.AlipayJSBridge) {  
    callback && callback();  
  } else {  
    // 如果没有注入则监听注入的事件  
    document.addEventListener('AlipayJSBridgeReady', callback, false);  
  }  
}  
ready(function() {  
  AlipayJSBridge.call('setOptionsMenu', {  
    title: '按钮',  
    redDot: '5', // -1 表示不显示, 0 表示显示红点, 1-99 表示在红点上显示的数字  
    color: '#ffff6600', // 必须以 # 开始 ARGB 颜色值  
  });  
  document.addEventListener('optionMenu', function(e) {  
    alert("optionMenu clicked!");  
  }, false);  
});  
</script>
```

多个 `optionMenu` 情况的事件：

```
<h1>请点击右上角每个菜单查看效果</h1>  
<script>  
function ready(callback) {  
  // 如果 jsbridge 已经注入则直接调用  
  if (window.AlipayJSBridge) {  
    callback && callback();  
  } else {  
    // 如果没有注入则监听注入的事件  
    document.addEventListener('AlipayJSBridgeReady', callback, false);  
  }  
}
```

```
ready(function() {
  AlipayJSBridge.call('setOptionsMenu', {
    // 显示的时候是从后往前显示的
    menus: [{
      icontype: 'scan',
      redDot: '-1', // -1 表示不显示, 0 表示显示红点, 1-99 表示在红点上显示的数字
    }, {
      icontype: 'user',
      redDot: '10', // -1 表示不显示, 0 表示显示红点, 1-99 表示在红点上显示的数字
    }],
    override: true //在需要设置多个 option 的情况下, 是否保留默认的 optionMenu
  });
  // 必须强制调用一把刷新界面
  AlipayJSBridge.call('showOptionsMenu');

  document.addEventListener('optionMenu', function(e) {
    alert(JSON.stringify(e.data));
  }, false);
});
</script>
```

注意事项

假如没有设置过 optionMenu, 那么这个事件是无法被触发的, 也就是说点击默认的 ... 是无法触发 optionMenu 事件的。

4.2.7 回退

用户点击导航栏左上角返回按钮或者 Android 设备的物理返回键时, 页面将会收到此事件 (back)。

如果在事件的处理函数中调用了 event.preventDefault(), 容器将忽略 backBehaviour, JS 需要负责回退或做其他操作。

注意：

- 对于 iOS 端侧滑返回, 由于手势操作具有可中途撤销的特性, 故无法触发 back 事件。
- 容器内部有保护机制, 如果用户多次重复点击返回按钮依然未能退出当前页, 将会触发强制退出, 忽略 preventDefault。

back 接口的使用方法

```
// 首先屏蔽默认行为, 然后调用页面跳转 API 进行手动控制
document.addEventListener('back', function(e) {
  e.preventDefault();
  console.log('do something...')
  AlipayJSBridge.call('popWindow');
}, false);
```

代码示例

点击回退跳转到指定页面：

```
<h1>请点击左上角返回按钮</h1>
<p>点击后会跳回到淘宝页面</p>
<script>
// 注意：如果自定义了 back，并使用了 location.href 去跳到指定的地址，需要包装一个 setTimeout 以保证不会阻塞客户端线程。
document.addEventListener('back', function(e) {
  e.preventDefault();
  setTimeout(function() {
    location.href = "https://m.taobao.com"
  }, 10);
}, false);
</script>
```

点击回退弹出确认框：

```
<h1>请点击左上角返回按钮</h1>
<script>
document.addEventListener('back', function(e) {
  e.preventDefault();

  AlipayJSBridge.call('confirm', {
    title: '亲',
    message: '确定要退出吗',
    okButton: '确定',
    cancelButton: '算了'
  }, function(e) {
    if (e.ok) {
      AlipayJSBridge.call('popWindow');
    }
  });
}, false);
</script>
```

4.2.8 添加通知

此接口用于添加 native 通知的监听。

addNotifyListener 接口的使用方法

```
AlipayJSBridge.call('addNotifyListener', {
  name:'fortest'
}, function (result) {
  console.log(result);
});
```

代码示例

以下示例为基本功能演示：

```
<h1>请点击下面的按钮来进行测试</h1>
<p>这里只测试在同一个页面内的情况，这个 API 可以在不同应用间通信</p>

<a href="#" class="btn start">开始监听</a>
<a href="#" class="btn stop">停止监听</a>
<a href="#" class="btn send">发通知</a>
<script>
function callback(e) {
alert(JSON.stringify(e));
};

function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}
ready(function(){
document.querySelector('.start').addEventListener('click', function() {
AlipayJSBridge.call('addNotifyListener', {
name:'NEBULANOTIFY_TEST_EVENT' // H5 发出的事件必须以 NEBULANOTIFY_ 开通进行监听
}, callback);
});

document.querySelector('.stop').addEventListener('click', function() {
AlipayJSBridge.call('removeNotifyListener', {
name:'NEBULANOTIFY_TEST_EVENT' // H5 发出的事件必须以 NEBULANOTIFY_ 开通进行监听
}, function(e) {
alert(JSON.stringify(e));
});
});

document.querySelector('.send').addEventListener('click', function() {
AlipayJSBridge.call('postNotification', {
name:'TEST_EVENT', // H5 发出的事件必须以 NEBULANOTIFY_ 开通进行监听
data: {
hello: 'world'
}
});
});
});
</script>
```

API

```
AlipayJSBridge.call('addNotifyListener', {
name, keep
}, fn)
```

入参

名称	类型	描述	必选	默认值
name	string	通知名称	Y	""
keep	string	是否在 ViewController 的生命周期内一直监听该通知	N	"true"
fn	function	回调函数	N	-

出参

result: {} : 回调函数带入的参数, 对应为事件的消息。

错误码

错误码	描述
4	无权限调用
12	在一个页面中多次监听同名通知

注意事项

- 在监听事件时, 如果这个事件是 H5 发出来的, 那么需要加上 NEBULANOTIFY_ 前缀。
- 在 addNotifyListener 时, 为了保险, 先做一次 remove 操作, 避免多次 add 报错。
- 如果 keep 设置为 false, 那么事件触发一次后自动反注册监听。
- Android 在回调中返回的 data 只有一层 key/value 键值对, 其中 value 如果在 postNotification 时是 Object, 则会被 Android 进行 JSON.stringify 操作, 使用时建议自行 try{JSON.parse}。

4.2.9 移除通知

该接口用于移除 native 通知的监听。

removeNotifyListener 接口的使用方法

```
AlipayJSBridge.call('removeNotifyListener', {
  name: 'fortest'
}, function (result) {
  console.log(result);
});
```

代码示例

以下示例为基本功能演示：

```
<h1>请点击下面的按钮来进行测试</h1>
<p>这里只测试在同一个页面内的情况, 这个 API 可以在不同应用间通信</p>

<a href="#" class="btn start">开始监听</a>
<a href="#" class="btn stop">停止监听</a>
<a href="#" class="btn send">发通知</a>
```

```

<script>
function callback(e){
alert(JSON.stringify(e));
};

function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}
ready(function(){
document.querySelector('.start').addEventListener('click', function() {
AlipayJSBridge.call('addNotifyListener', {
name:'NEBULANOTIFY_TEST_EVENT' // H5 发出的事件必须以 NEBULANOTIFY_ 开通进行监听
}, callback);
});

document.querySelector('.stop').addEventListener('click', function() {
AlipayJSBridge.call('removeNotifyListener', {
name:'NEBULANOTIFY_TEST_EVENT' // H5 发出的事件必须以 NEBULANOTIFY_ 开通进行监听
}, function(e) {
alert(JSON.stringify(e));
});
});

document.querySelector('.send').addEventListener('click', function() {
AlipayJSBridge.call('postNotification', {
name:'TEST_EVENT', // H5 发出的事件必须以 NEBULANOTIFY_ 开通进行监听
data: {
hello: 'world'
}
});
});
});
});
</script>

```

API

```

AlipayJSBridge.call('removeNotifyListener', {
name
}, fn)

```

入参

名称	类型	描述	必选	默认值
name	string	通知名称	Y	""
fn	function	回调函数	N	-

出参

result: {success} : 回调函数带入的参数。

名称	类型	描述
success	bool	是否成功移除

错误码

错误码	描述
4	无权限调用

注意事项

- 移除操作无论是否注册，都会返回 success : true。

4.2.10 分发消息

前端可以通过此接口给客户端发送通知，H5 会统一给传入的 name 标记加一个前缀 NEBULANOTIFY_，然后作为通知名称进行发送，如果您对这个通知感兴趣，监听的时候需要加前缀 NEBULANOTIFY_。

在 Android 10.1.0 之前的版本中，postNotification 的 data 字段不得为空，addNotifyListener 才可以接受。而在 10.1.0 版本中，此限制已被取消，addNotifyListener 可以接受一个空数据。

Android 是通过 LocalBroadcastManager 发送的广播，可以通过监听 NEBULANOTIFY_ + name 来进行监听。

postNotification 接口的使用方法

```
AlipayJSBridge.call('postNotification', {
  name:'fortest',
  data:{}
}, function (result) {
  console.log(result);
});
```

代码示例

以下示例为基本功能演示：

```
<h1>请点击下面的按钮来进行测试</h1>
<p>这里只测试在同一个页面内的情况，这个 API 可以在不同应用间通信</p>

<a href="#" class="btn start">开始监听</a>
<a href="#" class="btn stop">停止监听</a>
<a href="#" class="btn send">发通知</a>
<script>
function callback(e){
  alert(JSON.stringify(e));
};
```

```
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}
ready(function(){
document.querySelector('.start').addEventListener('click', function(){
AlipayJSBridge.call('addNotifyListener', {
name:'NEBULANOTIFY_TEST_EVENT' // H5 发出的事件必须以 `NEBULANOTIFY_` 开通进行监听
}, callback);
});

document.querySelector('.stop').addEventListener('click', function(){
AlipayJSBridge.call('removeNotifyListener', {
name:'NEBULANOTIFY_TEST_EVENT' // H5 发出的事件必须以 `NEBULANOTIFY_` 开通进行监听
}, function(e){
alert(JSON.stringify(e));
});
});

document.querySelector('.send').addEventListener('click', function(){
AlipayJSBridge.call('postNotification', {
name:'TEST_EVENT', // H5 发出的事件必须以 `NEBULANOTIFY_` 开通进行监听
data: {
hello: 'world'
}
}, function(e){
alert(JSON.stringify(e));
});
});
});
</script>
```

API

```
AlipayJSBridge.call('postNotification', {
name, data
}, fn)
```

入参

名称	类型	描述	必选	默认值
name	string	通知名称	Y	"
data	object	通知给客户端带的信息，Android 下会把 json 数据遍历，然后把里面每一项的 value 都转成 string 类型发送，请注意兼容数据格式。	N	-
fn	funct	回调函数	N	-

	ion		
--	-----	--	--

出参

result: {success} : 回调函数带入的参数。

名称	类型	描述
success	bool	是否消息发送成功

错误码

错误码	描述
4	无权限调用

4.3 页面上下文

4.3.1 打开新页面

pushWindow 用于在同一个离线包内打开一个新的页面，打开时自带系统转场动画。若您需要跨 appId 打开其他离线应用页面，请使用 startApp 接口。

pushWindow 与前端 location.href 的区别，类同于 PC 浏览器的新开标签页，每个 window 都是一个新的标签页，因此原页面仅仅是被压到后台，状态始终保持，JS 也会继续运行。

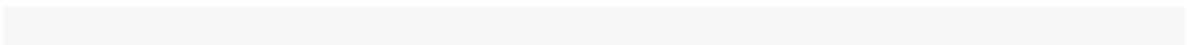
pushWindow 接口的使用方法

```
// 打开淘宝首页，自动读取 title，并且去除右边菜单按钮
AlipayJSBridge.call('pushWindow', {
  url: 'https://m.taobao.com/', // 要打开页面的 URL
  // 打开页面的配置参数
  param: {
    readTitle: true, //自动读取 title
    showOptionsMenu: false // 隐藏右边菜单
  },
  // 用于给新开的页面传递参数，可选
  // 在新开的页面使用 AlipayJSBridge.startupParams 可以获取到 passData
  passData: {
    key1:"key1Value",
    key2:"key2Value"
  }
});
```

注意：对于 Android 应用，需要将参数放在 param: {} 中；对于 iOS 应用，则需要将参数放在 passData: {} 里。

代码示例

打开淘宝首页，去除右边菜单：



```

<h1>打开淘宝首页</h1>
<a class="btn J_demo">执行</a>
<script>
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}
ready(function(){
document.querySelector('a').addEventListener('click', function() {
// 打开淘宝首页，自动读取 title，并且去除右边菜单
AlipayJSBridge.call('pushWindow', {
url: 'https://m.taobao.com/',
param: {
readTitle: true,
showOptionsMenu: false
}
});
});
});
});
</script>

```

打开时设置透明标题栏：

```

<h1>打开淘宝首页</h1>
<a class="btn J_demo">执行</a>
<script>
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}
ready(function() {
document.querySelector('a').addEventListener('click', function() {
AlipayJSBridge.call('pushWindow', {
url: 'https://m.taobao.com',
param: {
transparentTitle: 'auto'
}
});
});
});
});
</script>

```

API

```
AlipayJSBridge.call('pushWindow', {
  url, param, passData
})
```

入参

名称	类型	描述	必选	默认值
url	string	要打开的 URL	Y	""
param	dictionary	支持的 key/value 对下面的表格	N	{}
passData (仅适用于 iOS)	dictionary	传递的参数, 在打开页面使用 AlipayJSBridge.startupParams 可以获取	N	{}

param 参数详解

名称	类型	描述	默认值
defaultTitle	string	默认标题, 在页面第一次加载之前显示在标题栏上	""
showLoading	boolean	是否在页面加载前显示全局菊花	false
readTitle	boolean	是否读取网页标题显示在 titleBar 上	true
pullRefresh	boolean	是否支持下拉刷新 只有本地文件允许设置为 true	false
allowsBounceVertical	boolean	页面是否支持纵向拽拉超出实际内容, android 只支持下拉 (显示出背景或者域名) 只有 .alipay.com/.alipay.net/ 本地文件允许设置为 false	true
bounceTopColor	int	下拉超出时, 顶部间缝颜色 (十进制, 例如: bc=16775138)	-
showTitleLoading	boolean	是否在 TitleBar 的标题左边显示小菊花	false
transparentTitle	boolean	(不能与 titleBarColor 同时使用) always/auto: 1、always: 当前页面上下滚动时, titlebar 一直全透明; 2、auto: 当页面往下滚动时, 透明度不断增加, 直到 80 pt 时变成完全透明, 此时页面再往上滚动则反之, 透明度不断减小直到回到顶部时变成完全不透明。 3、none: 如果这个页面不需要透明效果, 则需要用 pushWindow 的 param 参数重新指定 transparentTitle 为 "none"	-
titleBarColor	int	自定义 titlebar 的背景色 (十进制, 例如: bc=16775138) 注意: 不能与 transparentTitle 同时使用	-
scrollDistance	int	在 transparentTitle 为 auto 的情况下, 滑动到透明度为 0.96 的距离	80 (iOS)
titleImage	string	所要使用的 title 图片地址, 请上传一张 3X PNG 图, 只影响当前的 VC, pushWindow 不会自动传递此参数, 为了更好的体验可以把图放在全局运营资源包中	""
closeCurrentWindow	boolean	打开窗口的同时, 关闭当前 window。	false
closeAllWindows	boolean	打开窗口的同时, 关闭当前 App 的所有 window	false
animationType	string	动画类型, 默认为 "push", 可用枚举 "none" / "push"	""

	ng	注意：android 未实现，均无动画	
--	----	---------------------	--

pushWindow 参数默认继承

名称	继承	备注
url	Y	-
defaultTitle	Y	-
backBehavior	Y	优先用户指定，否则 pop
showLoading	Y	-
readTitle	Y	-
pullRefresh	Y	-
toolbarMenu	Y	-
showProgress	Y	-
defaultSubtitle	Y	-
backgroundColor	Y	-
canPullDown	Y	-
showOptionsMenu	Y	-
showTitleLoading	Y	-
showDomain	Y	-

pushWindow 和 location.href 的区别

您可以如下理解 pushWindow 和 location.href 的区别：

- 首先把 Nebula 容器理解为一个 PC 浏览器。
- window 可以理解为标签页；location.href 就是正常的标签跳转。
- pushWindow 实际上就是新开了一个标签页，并且把之前那一页压到下面，push 出来的这页放在顶层展现。此时被压到下面的上一页所有状态均保留。
- location.href 就是依然停留在这个标签页，直接做页面跳转。
- 当 pushWindow 出来的标签页被关闭（pop）时，如果之前还有 window 存在，那么之前那个 window 就会恢复显示，并触发 resume。

注意事项

- pushWindow 打开 URL 页面的时候不会关闭已经存在的页面，注意打开个数，不要开太多影响性能。建议同一个应用 pushWindow 层级不要超过 5 层，否则会影响用户体验而且有可能导致应用 crash。

4.3.2 关闭当前页面

此接口用来关闭当前页面。

popWindow 接口的使用方法

```
// 关闭当前打开的页面
AlipayJSBridge.call('popWindow');
```

代码示例

关闭当前页面：

```
<h1>关闭当前页面</h1>
<a href="#" class="btn J_demo">执行</a>
<script>
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}
ready(function(){
document.querySelector('a').addEventListener('click', function() {
AlipayJSBridge.call('popWindow');
});
});
</script>
```

关闭当前页面并且传递数据：

```
<h1>点击"新开口", 然后点击"回退窗口"查看效果</h1>
<a href="#" class="btn pop">回退窗口</a>
<a href="#" class="btn new">新开口</a>
<script>
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}
ready(function() {
document.querySelector('.new').addEventListener('click', function() {
AlipayJSBridge.call('pushWindow', {
url: location.pathname
});
});
});

document.querySelector('.pop').addEventListener('click', function() {
AlipayJSBridge.call('popWindow', {
```

```

data: {
  from: location.href,
  info: Date.now()
}
});
});

document.addEventListener('resume', function(event) {
  alert('页面回退时带过来的内容 : ' + JSON.stringify(event.data));
});
});
</script>

```

API

```

AlipayJSBridge.call('popWindow', {
  data
})

```

入参

名称	类型	描述	必选	默认值
data	object	传递给当前 App 内即将露出页面的内容，无法跨 appId 传递数据。	N	-

注意事项

- 关于 popWindow 时所带的 data 如何被接收，请查看 页面恢复运行（resume 事件）。

4.3.3 关闭多个页面

此接口用于一次回退多级页面。

注意：只允许 popTo 到当前 App 实例内的页面，不允许跨 appId 跳转。

popTo 接口的使用方法

```

// 关闭当前打开的页面
AlipayJSBridge.call('popTo', {
  index: -1
});

```

代码示例

关闭当前页面并且传递数据：

```
<h1>点击"执行"关闭当前页面并返回数据</h1>
```

```

<a href="#" class="btn J_demo">执行</a>
<script>
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}
ready(function(){
document.querySelector('.J_demo').addEventListener('click', function() {
// 传递的 data 对象将会被即将露出的页面通过 resume 事件接收
AlipayJSBridge.call('popTo', {
index: -1, // 回退到上一个页面，假如这个时候没有上一个页面，就会报错
data: { // 特别强调：data 是字典，不是数组
from: location.href,
info: Date.now()
}
}, function(e) { // 添加回调，因为 popTo 不一定会成功（当前页面是唯一打开的页面的时候，会报错）
alert(JSON.stringify(e));
});
});
});
});
</script>

```

通过 urlPattern 返回到符合正则匹配的页面：

```

<h1>返回到符合某个规则的 URL</h1>
<h3></h3>
<a href="javascript:void(0)" class="btn J_new_window">新窗口打开当前页面</a>
<a href="javascript:void(0)" class="btn J_demo">返回</a>
<script>
var query = getQuery();
var depth = (+query.depth) || 0;
document.querySelector('h3').innerHTML = '当前页面深度: ' + depth;
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}
ready(function(){
document.querySelector('.J_demo').addEventListener('click', function() {
AlipayJSBridge.call('popTo', {
urlPattern: 'pop-to-url-pattern.html',
}, function(e) {
alert(JSON.stringify(e));
});
});
});

```

```
document.querySelector('.J_new_window').addEventListener('click', function() {
  AlipayJSBridge.call('pushWindow', {
    url: location.pathname + '?depth=' + (1+depth),
  });
});
});
});
</script>
```

API

```
AlipayJSBridge.call('popTo',{
  index, urlPattern
}, fn)
```

index、urlPattern 是 2 种查询模式，不应该被同时使用。

入参

名称	类型	描述	必选	默认值
index	int	目标界面在会话界面栈中的索引；如果小于零，则将与当前界面的 index 相加	Y	-
urlPattern	string	目标界面的 URL 匹配表达式（URL 如果包含 urlPattern，匹配成功）	Y	""
fn	function	操作成功时，回调可能不被调用；操作失败时，回调函数执行	N	-

出参

名称	类型	描述
result	undefined	操作成功时，回调可能不被调用；result 不应被使用。

错误码描述

错误码	描述
10	未配置参数； 无效的 index； 未匹配 urlPattern。

注意事项

- 一般情况下，popTo 用于分多步才能完成的场景，比如用户信息完善时回退，三级地址选择后返回等。
- 如果通过 urlPattern 来返回，popTo 会返回到离当前页面最远的页面，也就是栈底。同时不会去检测当前页面的 URL 是否符合。
- 关于 popTo 时所带的数据如何被接收，请查看 页面恢复运行（resume 事件）。

4.3.4 启动其他应用

此接口用于跨包打开 mPaaS 应用内的其他 H5 应用（离线包）。

startApp 接口的使用方法

```
AlipayJSBridge.call('startApp', {
  appId: '90000000',
  param: {
    url: '/index.html'
  }
}, function(result) {
  // noop
});

// 注意，如果要打开多个 App 实例：
// 请将 appClearTop 和 startMultApp 都放在 param 里
AlipayJSBridge.call('startApp', {
  appId: '90000000',
  param: {
    url: location.href,
    appClearTop: false,
    startMultApp: 'YES' // 注意这个值是 YES，而不是 bool 类型
  }
}, function(result) {
  // noop
});
```

代码示例

打开标题栏透明的应用：

```
<h1>点击按钮查看效果</h1>
<a href="javascript:void(0)" class="btn dream">打开心愿储蓄</a>
<script>
function ready(callback) {
  // 如果 jsbridge 已经注入则直接调用
  if (window.AlipayJSBridge) {
    callback && callback();
  } else {
    // 如果没有注入则监听注入的事件
    document.addEventListener('AlipayJSBridgeReady', callback, false);
  }
}
ready(function(){
  document.querySelector('.dream').addEventListener('click', function() {
    AlipayJSBridge.call('startApp', {
      appId: '20000981',
      param: {
        url: '/www/dream-create.html',
        // 启动参数传入
        canPullDown: true,
        transparentTitle: 'auto'
      }
    }, function(result) {
```

```
// noop
});
});
});
});
</script>
```

打开新应用并关闭当前应用：

```
<h1>点击按钮打开新应用，当前应用会被关闭</h1>
<a href="javascript:void(0)"class="btn forest">打开蚂蚁森林</a>
<script>
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}
ready(function() {
document.querySelector('.forest').addEventListener('click', function() {
AlipayJSBridge.call('startApp', {
appId: '60000002',
// 不传入 URL，就会使用 App 默认配置的 URL
param: {
transparentTitle: 'auto'
},
closeCurrentApp: true
}, function(result) {
// noop
});
});
});
});
});
</script>
```

默认只开一个 appId 实例：

```
<h1>尝试再打开当前页面，先退出当前应用，然后再次打开</h1>
<a href="javascript:void(0)"class="btn self">打开当前页面</a>
<script>
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}
ready(function(){
document.querySelector('.self').addEventListener('click', function() {
AlipayJSBridge.call('startApp', {
```

```
// 当前页面打开的时候，是通过通用应用 20000067 打开，
// 因此在此 startApp 的时候，就会把其他的 20000067 关闭
// 所以这个时候其实还是只有一个当前页面打开
appId: '20000067',
param: {
  url: location.href,
}
}, function(result) {
// noop
});
});
});
</script>
```

同一个 appId 多开：

```
<h1>打开多个相同 appId 的应用</h1>
<a href="javascript:void(0)"class="btn multi">再开启一个应用</a>
<script>
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
  callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}
ready(function() {
document.querySelector('.multi').addEventListener('click', function() {
AlipayJSBridge.call('startApp', {
  appId: '90000000',
  param: {
    url: '/index.html',
    appClearTop: false,
    startMultApp: 'YES' // 注意这个值是 YES，而不是 bool 类型
  }
}, function(result) {
// noop
});
});
});
</script>
```

API

```
AlipayJSBridge.call('startApp', {
  appId, param: {}, closeCurrentApp
}, fn)
```

入参

名称	类型	描述	必选	默认值	基线
----	----	----	----	-----	----

appId	string	离线包 ID。	Y	""	-
param	dictionary	启动应用的参数,由具体业务应用定义。如打开离线包时需指定url,请在 param 中配置。	N	value 值支持字符、bool、int、double	-
closeCurrentApp	bool	是否先退出当前 App 再启动新的 App。适用于页面用作中转页的情况。	N	-	>10.1.60
fn	function	调用失败后的回调函数。	N	-	-

错误码描述

错误码	描述
10	指定 appId 无效
11	启动 App 失败

注意事项

- startApp 是用来打开 App 的,因此其定位是 App 级别,在这一点上与 pushWindow 不同。
- 默认情况下,如果当前 App 已经被打开了,再次使用 startApp 打开 App 时,会进行一个类似重启的操作。也就是说不会出现同时运行两个具有相同 appId 的实例的情况。
- 假如一个 appId 要运行多个实例,那么请在 param 参数中添加 appClearTop=false&startMultApp=YES 选项。

4.3.5 退出当前应用

此接口用于退出当前栈顶 app。

exitApp 接口的使用方法

```
AlipayJSBridge.call('exitApp');
```

代码示例

退出当前的页面：

```
<h1>点击退出当前页面</h1>
<a href="#" class="btn J_demo">执行</a>
<script>
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
```

```
}  
}  
ready(function() {  
  document.querySelector('.J_demo').addEventListener('click', function() {  
    AlipayJSBridge.call('exitApp');  
  });  
});  
</script>
```

页面跳转完整实例：

```
<h1>请点击下面按钮来进行页面间跳转</h1>  
<h3></h3>  
<a href="javascript:void(0)"class="btn new">新开当前页面</a>  
<a href="javascript:void(0)"class="btn back">返回一级</a>  
<a href="javascript:void(0)"class="btn popTo">通过 popTo 退 2 级</a>  
<a href="javascript:void(0)"class="btn exit">关闭所有页面</a>  
<script>  
var query = getQuery();  
var depth = (+query.depth) || 0;  
document.querySelector('h3').innerHTML = '当前页面深度: ' + depth;  
function ready(callback) {  
  // 如果 jsbridge 已经注入则直接调用  
  if (window.AlipayJSBridge) {  
    callback && callback();  
  } else {  
    // 如果没有注入则监听注入的事件  
    document.addEventListener('AlipayJSBridgeReady', callback, false);  
  }  
}  
ready(function() {  
  document.querySelector('.new').addEventListener('click', function() {  
    AlipayJSBridge.call('pushWindow', {  
      url: location.pathname + '?depth=' + (1+depth),  
    });  
  });  
  document.querySelector('.back').addEventListener('click', function() {  
    AlipayJSBridge.call('popWindow', {  
      data: {  
        method: 'popWindow',  
        depth: depth,  
      }  
    });  
  });  
  document.querySelector('.popTo').addEventListener('click', function() {  
    AlipayJSBridge.call('popTo', {  
      index: -2,  
      data: {  
        method: 'popTo',  
        depth: depth,  
      }  
    }, function(e) {  
      if (e.error) {  
        alert('发生错误: ' + JSON.stringify(e));  
      }  
    }  
  }  
});
```

```

});
});
document.querySelector('.exit').addEventListener('click', function() {
AlipayJSBridge.call('exitApp');
});
});
document.addEventListener('resume', function(event) {
alert('页面回退时带过来的内容 : ' + JSON.stringify(event.data));
});
</script>

```

API

```

AlipayJSBridge.call('exitApp', {
closeActionType, animated
}, fn);

```

入参

名称	类型	描述	必选	默认值
closeActionType	string	exitSelf 退出自身应用； exitTop 退出栈顶应用	N	-
animated	bool	是否开启动画	N	true

注意事项

- 对于没有申请过 appId 的页面，都是以 20000067 这个 appId 运行，因此任何页面内调用 exitApp，所有页面都会关闭。

4.3.6 startH5App

启动 H5 应用

此方法目前已经 deprecated, 请使用 startApp。

startH5App 接口的使用方法

```

AlipayJSBridge.call('startH5App', {
appId: '20000067',
url: '/www/index.html',
});

```

代码示例

```

<h1>点击按钮查看效果</h1>
<a href="javascript:void(0)"class="btn dream">打开心愿储蓄</a>
<script>

```

```
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}
ready(function(){
document.querySelector('.dream').addEventListener('click', function() {
AlipayJSBridge.call('startH5App', {
appId: '20000981',
url: '/www/dream-create.html',
// 启动参数传入
});
});
});
</script>
```

API

```
AlipayJSBridge.call('startH5App', {
appId, url, ...param
})
```

入参

名称	类型	描述	必选
appId	string	应用 H5 appId	Y
url	string	启动应用的参数, 由具体业务应用定义	N
param	object	k-v 形式, 不支持嵌套	N

4.4 Native 功能

4.4.1 扫码解析

此接口用于调用扫码组件, 且仅限于 Android 系统。另外, 在使用此接口前, 请确认您已经在工程中添加了扫码组件。

三种不同的 actionType 为 :

- 获取码值
- 用码值路由
- 获取码值并用码值路由

scan 接口的使用方法

```
AlipayJSBridge.call('scan', {
  type: 'bar',
  actionType: 'scanAndRoute'
}, function(result) {
  alert(JSON.stringify(result));
});
```

代码示例

只获取二维码的信息：

```
<h1>点击扫码后输出码对应的信息</h1>
<a href="#" class="btn read">开始扫码</a>
<script>
function ready(callback) {
  // 如果 jsbridge 已经注入则直接调用
  if (window.AlipayJSBridge) {
    callback && callback();
  } else {
    // 如果没有注入则监听注入的事件
    document.addEventListener('AlipayJSBridgeReady', callback, false);
  }
}
ready(function() {
  document.querySelector('.read').addEventListener('click', function() {
    AlipayJSBridge.call('scan', {
      type: 'qr'
    }, function(result) {
      alert(JSON.stringify(result));
    });
  });
});
</script>
```

扫描二维码并发起路由：

```
<h1>扫码后不输出信息，直接让支付宝解码</h1>
<a href="#" class="btn read">开始扫码</a>
<script>
function ready(callback) {
  // 如果 jsbridge 已经注入则直接调用
  if (window.AlipayJSBridge) {
    callback && callback();
  } else {
    // 如果没有注入则监听注入的事件
    document.addEventListener('AlipayJSBridgeReady', callback, false);
  }
}
ready(function() {
  document.querySelector('.read').addEventListener('click', function() {
    AlipayJSBridge.call('scan', {
      type: 'qr',
```

```

actionType: 'scanAndRoute'
}, function(result) {
alert(JSON.stringify(result));
});
});
});
</script>

```

API

```

AlipayJSBridge.call('scan', {
type, actionType, qrCode
}, fn);

```

入参

名称	类型	描述	必选	默认值
type	string	扫描目标类型，二维码/条形码。	Y	""
actionType	string	操作类型： "scan" 只扫码值。 "route" 用码值路由。 "scanAndRoute" 扫描并路由，类同首页效果。	N	"scan"
qrCode	string	指定用于 "route" 操作类型的码值。	N	""
fn	function	扫码获取码信息后的回调函数。	N	-

出参

回调函数带入的参数 result: {error, barCode, qrCode, cardNumber}。

名称	类型	描述
barCode	string	扫描所得条码数据。
qrCode	string	扫描所得二维码数据。
error	int	错误码（10：用户取消，11：操作失败）

4.5 界面

4.5.1 警告框

该接口用于警告框的 native 实现。

alert 接口的使用方法

```

AlipayJSBridge.call('alert', {

```

```
title: '亲',
message: '你好',
button: '确定'
}, function(e) {
alert(JSON.stringify(e));
});
```

代码示例

- alert 和 confirm :

```
<h1>点击以下按钮看不同效果</h1>
<a href="javascript:void(0)"class="btn alert">点击 Alert</a>
<a href="javascript:void(0)"class="btn confirm">点击 Confirm</a>
<script>
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}
ready(function() {
document.querySelector('.alert').addEventListener('click', function() {
AlipayJSBridge.call('alert', {
title: '亲',
message: '你好',
button: '确定'
}, function(e) {
alert(JSON.stringify(e));
});
});
document.querySelector('.confirm').addEventListener('click', function() {
AlipayJSBridge.call('confirm', {
title: '亲',
message: '确定要退出吗?',
okButton: '是',
cancelButton: '否'
}, function(e) {
alert(JSON.stringify(e));
});
});
});
});
</script>
```

API

```
AlipayJSBridge.call('alert',{
title, message, button
}, fn)
```

入参

名称	类型	描述	必选	默认值
title	string	Alert 框标题	N	""
message	string	Alert 框文本	N	""
align	string	message 对齐方式, 可用枚举 left/center/right	N	iOS "center", Android "left"
button	string	按钮文字	N	"确定"
fn	function	回调函数, 当点击 button 后被调用	N	-

注意事项

- 与 window.alert 不同的是, alert 不是阻塞式的, 也就是说如果先后弹了 2 个警告框, 最后看到的是后弹的那个。

4.5.2 确认框

确认框的 native 实现。

confirm 接口的使用方法

```
AlipayJSBridge.call('confirm', {
  title: '亲',
  message: '确定要退出吗?',
  okButton: '是',
  cancelButton: '否'
}, function(e) {
  alert(JSON.stringify(e));
});
```

代码示例

alert 和 confirm :

```
<h1>点击以下按钮看不同效果</h1>
<a href="javascript:void(0)"class="btn alert">点击Alert</a>
<a href="javascript:void(0)"class="btn confirm">点击Confirm</a>
<script>
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}
ready(function() {
document.querySelector('.alert').addEventListener('click', function() {
AlipayJSBridge.call('alert', {
title: '亲',
```

```

message: '你好',
button: '确定'
}, function(e) {
e && alert(JSON.stringify(e))
});
});
document.querySelector('.confirm').addEventListener('click', function(){
AlipayJSBridge.call('confirm', {
title: '亲',
message: '确定要退出吗?',
okButton: '是',
cancelButton: '否'
}, function(e) {
alert(JSON.stringify(e))
});
});
});
</script>

```

API

```

AlipayJSBridge.call('alert',{
title, message, okButton, cancelButton
}, fn)

```

入参

名称	类型	描述	必选	默认值
title	string	Alert 框标题	N	""
message	string	Alert 框文本	N	""
align	string	message 对齐方式, 可用枚举 left/center/right	N	iOS "center", Android "left"
okButton	string	确定按钮文字	N	"确定"
cancelButton	string	取消按钮文字	N	"取消"
fn	function	回调函数, 当点击 button 后被调用	N	-

注意事项

- 与 alert 相同, confirm 也不是阻塞式的, 也就是说如果先后弹了 2 个警告框, 最后看到的是后弹的那个。

4.5.3 弱提示

此接口用于显示一个弱提示, 可选择多少秒之后消失。

toast 接口的使用方法

```
AlipayJSBridge.call('toast', {
  content: '操作成功',
  type: 'success',
  duration: 2000
}, function() {
  alert("toast消失后执行");
});

// 可以通过 hideToast 接口隐藏已经弹出的 toast

AlipayJSBridge.call('hideToast', {}, function() {
});
```

代码示例

```
<h1>点击以下按钮看不同效果</h1>
<a href="javascript:void(0)" class="btn success">显示成功信息</a>
<a href="javascript:void(0)" class="btn fail">显示失败信息</a>
<a href="javascript:void(0)" class="btn exception">显示异常信息</a>
<a href="javascript:void(0)" class="btn none">只显示信息</a>
<a href="javascript:void(0)" class="btn duration">3.5s信息显示</a>

<script>
function toast(config, callback){
  AlipayJSBridge.call('toast',config, callback);
}

function ready(callback) {
  // 如果 jsbridge 已经注入则直接调用
  if (window.AlipayJSBridge) {
    callback && callback();
  } else {
    // 如果没有注入则监听注入的事件
    document.addEventListener('AlipayJSBridgeReady', callback, false);
  }
}

ready(function() {
  document.querySelector('.success').addEventListener('click', function() {
    toast({
      content: '操作成功',
      type: 'success'
    });
  });

  document.querySelector('.fail').addEventListener('click', function() {
    toast({
      content: '网络繁忙，请稍后再试',
      type: 'fail'
    });
  });

  document.querySelector('.exception').addEventListener('click', function() {
    toast({
```

```

content: '发生异常，请注意',
type: 'exception'
});
});

document.querySelector('.none').addEventListener('click', function() {
toast({
content: '欢迎光临',
});
});

document.querySelector('.duration').addEventListener('click', function() {
toast({
content: '请稍等',
duration: 3500,
}, function(e){
alert('toast消失回调');
});
});
});
});
</script>

```

API

```

AlipayJSBridge.call('toast', {
content, type, duration
}, fn)

```

入参

名称	类型	描述	必选	默认值
content	string	文字内容。	Y	""
type	string	none / success / fail / exception。 excption 类型必须传文案。	N	"none"
duration	int	显示时长，单位为毫秒。	N	2000
xOffset	float	左为正方向，单位为 px。	N	0
yOffset	float	上为正方向，单位为 px。	N	0
fn	function	回调函数，当 toast 消失后被调用。	N	-

注意事项

- toast 虽然会自动关闭，但是也可以通过 hideLoading 来关闭，这种使用方式不常见，但是要防止被 hideLoading 关闭。
- 对于 Android 系统，需要注意的是，如果系统通知关闭，则此弹框不会出现。
- Android 10.1.2 以下版本中，duration 显示时长只支持 2000/3500 两种，小于或等于 2000 的相当于 2000, 大于 2000 的相当于 3500。

4.5.4 选择列表

此接口提供一种供选择选项的列表，它会停留在屏幕的下边沿。

actionSheet 接口的使用方法

```
AlipayJSBridge.call('actionSheet', {
  'title': '标题',
  'btns': ['第一个按钮', '第二个按钮', '第三个按钮'],
  'cancelBtn': '取消',
  'destructiveBtnIndex': 2
}, function(data) {
  switch (data.index) { // index 标示用户点击的按钮在 actionSheet 中的位置，从 0 开始
    case 0:
      alert('第一个按钮');
      break;
    case 1:
      alert('第二个按钮');
      break;
    case 2:
      alert('第三个按钮');
      break;
    case 3:
      alert('取消按钮');
      break;
  }
});
```

代码示例

```
<h1>点击按钮调出 actionSheet</h1>
<a href="javascript:void(0)" class="btn actionSheet">打开 actionSheet</a>

<script>
function ready(callback) {
  // 如果 jsbridge 已经注入则直接调用
  if (window.AlipayJSBridge) {
    callback && callback();
  } else {
    // 如果没有注入则监听注入的事件
    document.addEventListener('AlipayJSBridgeReady', callback, false);
  }
}

ready(function() {
  document.querySelector('.actionSheet').addEventListener('click', function() {
    AlipayJSBridge.call('actionSheet',{
      'title': '标题',
      'btns': ['第一个按钮', '第二个按钮', '第三个按钮'],
      'cancelBtn': '取消',
      'destructiveBtnIndex': 2
    }, function(data) {
      switch (data.index) { // index 标示用户点击的按钮，在 actionSheet 中的位置，从 0 开始
        case 0:
          alert('第一个按钮');
          break;
```

```

case 1:
alert('第二个按钮');
break;
case 2:
alert('第三个按钮');
break;
case 3:
alert('取消按钮');
break;
}
});
});
});
</script>

```

API

```

AlipayJSBridge.call('actionSheet',{
title, btns, cancelBtn, destructiveBtnIndex
}, fn)

```

入参

名称	类型	描述	必选	默认值
title	string	标题	N	''
btns	array	一组按钮，item 类型是 string。	Y	[]
cancelBtn	string	设置取消按钮及文字。	N	''
destructiveBtnIndex	int	(iOS 特殊处理) 指定按钮的索引号，从 0 开始； 使用场景：需要删除或清除数据等类似场景，默认红色。	N	-
fn	function	不是 API 调用后被回调，而是用户选择选项之后的回调函数。	N	-

出参

格式为 {data: {index: 0}}。其中 index 是用户点击的按钮在 actionSheet 中的位置，从 0 开始。

4.5.5 设置标题

此接口用于设置页面的标题栏，包括主标题、副标题以及标题菜单项。

注意：由于苹果的 ATS 限制，image URL 必须为 https 链接或 base64，而 http 链接会被忽略。

setTitle 接口的使用方法

```
AlipayJSBridge.call('setTitle', {  
  title: '标题',  
});
```

代码示例

设置各种类型的标题栏：

```
<h1>点击以下按钮看不同效果</h1>  
<a href="javascript:void(0)" class="btn title">只设置标题</a>  
<a href="javascript:void(0)" class="btn subTitle">标题+副标题</a>  
<a href="javascript:void(0)" class="btn clear">清空标题</a>  
  
<script>  
function ready(callback) {  
  // 如果 jsbridge 已经注入则直接调用  
  if (window.AlipayJSBridge) {  
    callback && callback();  
  } else {  
    // 如果没有注入则监听注入的事件  
    document.addEventListener('AlipayJSBridgeReady', callback, false);  
  }  
}  
  
ready(function(){  
  document.querySelector('.title').addEventListener('click', function() {  
    AlipayJSBridge.call('setTitle', {  
      title: '标题'  
    });  
  });  
});  
  
document.querySelector('.subTitle').addEventListener('click', function() {  
  AlipayJSBridge.call('setTitle', {  
    title: '标题',  
    subtitle: '副标题'  
  });  
});  
  
document.querySelector('.clear').addEventListener('click', function() {  
  AlipayJSBridge.call('setTitle', {  
    title: '',  
    subtitle: '',  
  });  
});  
});  
</script>
```

API

```
AlipayJSBridge.call('setTitle',{  
  title, subtitle, image  
}, fn)
```

入参

名称	类型	描述	必选	默认值
title	string	主标题文案	N	""
subtitle	string	副标题文案	N	""
image	string	支持 URL 或者 base64，请使用一张 3X 图，如果设置了 image，则前两个参数失效，并且不从 webview 的回调中读取 title。	N	""

注意事项

安卓 10.0.18 版本之前不支持设置空的 title，可以通过设置一个不可见字符串绕过，10.0.20 版本已经去掉这个限制。

```
AlipayJSBridge.call('setTitle', {
  title: '\u200b',
});
```

4.5.6 设置导航栏底部细线颜色

此接口用于自定义导航栏底部细线的颜色，当细线的颜色与导航栏相同时可达到隐藏的效果。

setBarBottomLineColor 接口的使用方法

```
AlipayJSBridge.call("setBarBottomLineColor", {
  color: 16711688
});
```

代码示例

```
<div style="padding-top:80px;">
<a href="javascript:void(0)" class="btn title">设置导航栏底部细线颜色</a><br>
</div>

<script>
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
  callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}

ready(function(){
document.querySelector('.title').addEventListener('click', function() {
AlipayJSBridge.call("setBarBottomLineColor", {
color: parseInt('ff0000', 16)
```

```
});
}, false);
},false);
</script>
```

API

```
AlipayJSBridge.call('setBarBottomLineColor',{
color
}, fn)
```

入参

名称	类型	描述	必选	默认值
color	int	色值，十进制	Y	-

4.5.7 设置导航栏背景色

此接口用于设置 TitleBar 的颜色。

setTitleColor 接口的使用方法

```
AlipayJSBridge.call("setTitleColor", {
color: 16775138,
reset: false // (可选，默认为 false) 是否重置 title 颜色为默认颜色。
});
```

代码示例

```
<div style="padding-top:80px;">
<a href="javascript:void(0)"class="btn title">设置导航栏背景色</a>
<a href="javascript:void(0)"class="btn reset">重置导航栏背景色</a>
<a href="javascript:void(0)"class="btn pushWindow">新开一个透明导航栏的窗口</a>
<a href="javascript:void(0)"class="btn resetTransparent">重置为透明导航栏</a>
</div>

<script>
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}
ready(function(){
document.querySelector('.title').addEventListener('click', function() {
AlipayJSBridge.call("setTitleColor", {
```

```

color: parseInt('108ee9', 16),
reset: false // ( 可选, 默认为 false ) 是否重置 title 颜色为默认颜色。
});
});

document.querySelector('.reset').addEventListener('click', function() {
AlipayJSBridge.call("setTitleColor",{
color: 16775138,
reset: true
});
});

document.querySelector('.pushWindow').addEventListener('click', function() {
AlipayJSBridge.call("pushWindow", {
url: location.pathname + '?_webview_options__=transparentTitle%3Dalways'
});
});

document.querySelector('.resetTransparent').addEventListener('click', function() {
AlipayJSBridge.call("setTitleColor", {
color: 16775138,
resetTransparent: true
});
});
});
});
</script>

```

API

```

AlipayJSBridge.call('setTitleColor', {
color: 16775138,
reset: false,
resetTransparent: false
}, fn)

```

入参

名称	类型	描述	必选	默认值
color	int	色值, 十进制	Y	-

4.5.8 设置右上角按钮

- 此接口用于设置标题栏右边的按钮属性。
- 此接口仅负责设置, 需要额外调用 showOptionsMenu 保证该按钮的显示。

注意: 由于苹果的 ATS 限制, icon URL 必须为 https 链接或 base64, 而 http 链接会被忽略。

setOptionsMenu 接口的使用方法

```
AlipayJSBridge.call('setOptionsMenu', {
  title: '按钮', // 与 icon、icontype 三选一
  redDot: '-1', // -1 表示不显示, 0 表示显示红点, 1-99 表示在红点上显示的数字
  color: '#ff00ff00', // 必须以 # 开始 ARGB 颜色值
});
```

代码示例

设置各种类型的右侧按钮：

```
<h1>点击以下按钮看不同效果</h1>

<a href="javascript:void(0)" class="btn button">单按钮</a>
<a href="javascript:void(0)" class="btn icon">单图标</a>
<a href="javascript:void(0)" class="btn menu">多菜单(9.9.3)</a>
<a href="javascript:void(0)" class="btn reset">重置</a>
<a href="javascript:void(0)" class="btn hide">隐藏</a>
<a href="javascript:void(0)" class="btn show">显示</a>

<script>
function ready(callback) {
  // 如果 jsbridge 已经注入则直接调用
  if (window.AlipayJSBridge) {
    callback && callback();
  } else {
    // 如果没有注入则监听注入的事件
    document.addEventListener('AlipayJSBridgeReady', callback, false);
  }
}

ready(function(e) {
  document.querySelector('.button').addEventListener('click', function() {
    AlipayJSBridge.call('setOptionsMenu', {
      title: '按钮',
      redDot: '5', // -1 表示不显示, 0 表示显示红点, 1-99 表示在红点上显示的数字
      color: '#ff00ff00', // 必须以 # 开始 ARGB 颜色值
    });
    AlipayJSBridge.call('showOptionsMenu');
  });

  document.querySelector('.icon').addEventListener('click', function() {
    AlipayJSBridge.call('setOptionsMenu', {
      icon: 'http://pic.alipayobjects.com/e/201212/1ntOVeWwtg.png',
      redDot: '-1', // -1 表示不显示, 0 表示显示红点, 1-99 表示在红点上显示的数字
    });
    AlipayJSBridge.call('showOptionsMenu');
  });

  document.querySelector('.menu').addEventListener('click', function() {
    AlipayJSBridge.call('setOptionsMenu', {
      // 显示时的顺序为从右至左
      menus: [{
        icontype: 'scan',
        redDot: '-1', // -1 表示不显示, 0 表示显示红点, 1-99 表示在红点上显示的数字
      }, {
```

```

icontype: 'user',
redDot: '-1', // -1 表示不显示, 0 表示显示红点, 1-99 表示在红点上显示的数字
}},
override: true // 在需要设置多个 option 的情况下, 是否保留默认的 optionMenu
});

// 必须强制调用一次以刷新界面
AlipayJSBridge.call('showOptionsMenu');
});

document.querySelector('.reset').addEventListener('click', function() {
AlipayJSBridge.call('setOptionsMenu', {
reset: true,
});
AlipayJSBridge.call('showOptionsMenu');
});

document.querySelector('.show').addEventListener('click', function() {
AlipayJSBridge.call('showOptionsMenu');
});

document.querySelector('.hide').addEventListener('click', function() {
AlipayJSBridge.call('hideOptionsMenu');
});

document.addEventListener('optionMenu', function(e) {
alert(JSON.stringify(e.data));
}, false);
});
</script>

```

API

要注意的是有几个属性是优先级的：reset > title > icontype > icon，这 4 个属性只需要有一个即可。

```

AlipayJSBridge.call('setTitle',{
title, icon, redDot, reset, color, override, menus, icontype
})

```

入参

名称	类型	描述	默认值
title	string	右按钮文字	""
icon	string	右按钮图标 URL, base64 (since 9.0) 8.3 及以前: iOS 40x40 (周边不留白), Android 50x50 (四边各透明留白 5px) 8.4 及以后: 两个平台统一使用 40x40 (周边不留白)	""

redDot	string	红点数值		""
reset	boolean	重置到系统默认, 当 reset=true 时, 忽略其他参数		false
color	string	文字颜色值		"#FFFFFF"
override	boolean	在需要设置多个 option 的情况下, 是否保留默认的 optionMenu		false
menus	array	设置多个按钮		[]
preventDefault	boolean	是否阻止默认的分享功能 (默认是弹分享框) preventDefault=true 会阻止默认的分享		[]
icontype	string	根据图片类型加载容器预置图片, 与 title、icon 三选一。 注意: 只支持单个 optionMenu 变色。 具体类型包含: user (账户)、filter (筛选器)、search (查找)、add (添加)、settings (设置)、scan (扫一扫)、info (信息)、help (帮助)、locate (定位)、more (更多)、mail (邮箱 10.0.8 及以上)		""
contentDesc	string	设置盲人模式阅读文案		""

注意事项

- setOptionMenu 后如果效果不对, 请调用一次 showOptionMenu。

4.5.9 显示右上角按钮

此接口用于显示标题栏右边按钮的属性。

showOptionMenu 接口的使用方法

```
AlipayJSBridge.call('showOptionMenu');
```

代码示例

请参考：设置右上角按钮 setOptionsMenu。

4.5.10 隐藏右上角按钮

此接口用于隐藏标题栏右边按钮的属性。

hideOptionsMenu 接口的使用方法

```
AlipayJSBridge.call('hideOptionsMenu');
```

代码示例

请参考：设置右上角按钮 setOptionsMenu。

4.5.11 显示加载中

此接口用于显示全局加载框。

showLoading 接口的使用方法

```
AlipayJSBridge.call('showLoading', {  
  text: '加载中',  
});
```

代码示例

显示/隐藏全局加载框：

```
<h1>点击以下按钮看不同效果</h1>  
<p>注意安卓下显示 loading 后，会覆盖整个界面，所以请使用系统回退键关闭 loading</p>  
<a href="javascript:void(0)"class="btn show">显示 loading</a>  
<a href="javascript:void(0)"class="btn delay">延迟 2 秒显示 loading</a>  
<a href="javascript:void(0)"class="btn notext">无文字菊花</a>  
  
<script>  
function ready(callback) {  
  // 如果 jsbridge 已经注入则直接调用  
  if (window.AlipayJSBridge) {  
    callback && callback();  
  } else {  
    // 如果没有注入则监听注入的事件  
    document.addEventListener('AlipayJSBridgeReady', callback, false);  
  }  
}  
ready(function() {  
  document.querySelector('.show').addEventListener('click', function() {  
    AlipayJSBridge.call('showLoading', {  
      text: 'Loading',  
    });  
  });  
});
```

```

setTimeout(function() {
  AlipayJSBridge.call('hideLoading');
}, 3000);
});

document.querySelector('.delay').addEventListener('click', function() {
  AlipayJSBridge.call('showLoading', {
    text: 'Loading',
    delay: 2000,
  });
  setTimeout(function() {
    AlipayJSBridge.call('hideLoading');
  }, 5000);
});

document.querySelector('.notext').addEventListener('click', function() {
  AlipayJSBridge.call('showLoading', {
    text: ''
  });
  setTimeout(function() {
    AlipayJSBridge.call('hideLoading');
  }, 3000);
});
});
</script>

```

API

```

AlipayJSBridge.call('showLoading',{
  text, delay
})

```

入参

名称	类型	描述	必选	默认值
text	string	文本内容；如需设为无文案，需传入一个空格	N	“加载中”
delay	int	延迟多少毫秒后显示；如果在此时间之前调用了 hideLoading，则不会再显示	N	0
autoHide	boolean	默认情况下容器会在 pageFinish 后主动隐藏加载框，默认 true，传入 false，关掉自动隐藏（only Android）	N	true
cancelable	boolean	安卓返回键是否消掉加载框，默认物理返回键会消掉加载框（only Android）	N	true

注意事项

- 安卓下显示 loading 后，会覆盖整个界面，所以请使用系统回退键关闭 loading。
- iOS 下的情况是，当没有设置 text 值的时候，只有标题栏和工具栏可以被点击，有文字的时候不能覆盖任何内容。9.9.5 以后已修复。
- showLoading 是 webview 级别的，因此在 pushwindow 后的 webview 上调用 hideLoading 无法关掉

上个 webview 的 loading，需要保证 showLoading 和 hideLoading 在一个 webview 环境下执行。

4.5.12 隐藏加载中

此接口用于隐藏全局加载框。

hideLoading 接口的使用方法

```
AlipayJSBridge.call('hideLoading');
```

代码示例

显示/隐藏全局加载框：

```
<h1>点击以下按钮看不同效果</h1>
<p>注意安卓下显示 loading 后，会覆盖整个界面，所以请使用系统回退键关闭 loading</p>
<button class="btn show">显示 loading</button>
<button class="btn delay">延迟 2 秒显示 loading</button>
<button class="btn notext">无文字菊花</button>

<script>
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}
ready(function() {
document.querySelector('.show').addEventListener('click', function() {
AlipayJSBridge.call('showLoading', {
text: 'Loading',
});
setTimeout(function() {
AlipayJSBridge.call('hideLoading');
}, 3000);
});

document.querySelector('.delay').addEventListener('click', function() {
AlipayJSBridge.call('showLoading', {
text: 'Loading',
delay: 2000,
});
setTimeout(function() {
AlipayJSBridge.call('hideLoading');
}, 5000);
});

document.querySelector('.notext').addEventListener('click', function() {
AlipayJSBridge.call('showLoading', {
```

```
text: ' ',
});
setTimeout(function() {
  AlipayJSBridge.call('hideLoading');
}, 3000);
});
});
</script>
```

4.5.13 显示标题栏加载中

此接口用于在标题栏显示加载框。

showTitleLoading 接口的使用方法

```
AlipayJSBridge.call('showTitleLoading');
```

代码示例

显示/隐藏全局加载框：

```
<h1>点击下面按钮查看效果</h1>
<a href="javascript:void(0)"class="btn show">显示 loading</a>
<a href="javascript:void(0)"class="btn hide">隐藏 loading</a>

<script>
function ready(callback) {
  // 如果 jsbridge 已经注入则直接调用
  if (window.AlipayJSBridge) {
    callback && callback();
  } else {
    // 如果没有注入则监听注入的事件
    document.addEventListener('AlipayJSBridgeReady', callback, false);
  }
}
ready(function() {
  document.querySelector('.show').addEventListener('click', function() {
    AlipayJSBridge.call('showTitleLoading');
  });

  document.querySelector('.hide').addEventListener('click', function() {
    AlipayJSBridge.call('hideTitleLoading');
  });
});
</script>
```

4.5.14 隐藏标题栏加载中

此接口用于隐藏标题栏加载框。

hideTitleLoading 接口的使用方法

```
AlipayJSBridge.call('hideTitleLoading');
```

代码示例

显示/隐藏全局加载框：

```
<h1>点击下面按钮查看效果</h1>
<a href="javascript:void(0)"class="btn show">显示 loading</a>
<a href="javascript:void(0)"class="btn hide">隐藏 loading</a>

<script>
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}
ready(function() {
document.querySelector('.show').addEventListener('click', function() {
AlipayJSBridge.call('showTitleLoading');
});

document.querySelector('.hide').addEventListener('click', function() {
AlipayJSBridge.call('hideTitleLoading');
});
});
</script>
```

4.6 工具类

4.6.1 获取容器的启动参数

此接口用于获取容器的启动参数（本 JS API 仅支持获取业务参数，对非业务参数或者缩写参数的传入不保证后果）。

getStartupParams 接口的使用方法

```
AlipayJSBridge.call('getStartupParams', {
key: ['url', 'xxx'] // 可选，根据 key 值过滤返回结果，不填返回全部
}, function(result) {
console.log(result);
});
```

代码示例

```

<script>
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}

ready(function() {
// 有 key 参数的情况
AlipayJSBridge.call('getStartupParams', {
key: ['url']
}, function(result) {
alert(JSON.stringify(result));
});

// 没有参数的情况
AlipayJSBridge.call('getStartupParams', function(result) {
alert(JSON.stringify(result));
});
});
</script>

```

API

getStartupParams

入参

名称	类型	描述	必选	默认值
key	Array	根据传的 key 来获得对应的 Key 的 value	N	null

出参

返回对应的启动参数，例如：{url: 'https://taobao.com', xx: '其他启动参数'}。

- 如果没有入参，则返回所有的 startupParams 参数。
- 如果有入参，根据入参返回对应的 value。
- 如果启动参数中没有对应 key 值，则返回中不带这个 key，不做报错。

错误码描述 (类型 Number)

错误码	描述
2	参数异常，key 为空数组，或者其它类型。
12	未知错误

4.6.2 截屏

此接口用于截屏。

snapshot 的使用方法

```
AlipayJSBridge.call('snapshot', function(result) {
  console.log(result.success);
});
```

代码示例

```
<h1>点击下面按钮查看不同截图效果</h1>

<a href="javascript:void(0)" class="btn screen">截取屏幕并保存到相册</a>
<a href="javascript:void(0)" class="btn viewport">viewport 截图返回 fileURL</a>

<script>
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
  callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}
ready(function() {
document.querySelector('.screen').addEventListener('click', function() {
AlipayJSBridge.call('snapshot', function(result) {
alert(JSON.stringify(result));
});
});

document.querySelector('.viewport').addEventListener('click', function() {
AlipayJSBridge.call('snapshot', {
  range: 'viewport',
  dataType: 'fileURL',
  saveToGallery: false
}, function(result) {
alert(JSON.stringify(result));
});
});
});
</script>
```

API

```
AlipayJSBridge.call('snapshot', {
  range, saveToGallery, dataType, imageFormat, quality,
  maxWidth, maxHeight
```

```
}, fn)
```

入参

名称	类型	描述	必选	默认值
range	string	快照范围： screen：当前客户端整个屏幕 viewport：网页可见区域 document：整个网页。 注意：document 会截取全部网页，在安卓手机上，网页很大的时候会浏览器内存溢出，请使用 screen	N	"screen"
saveToGallery	bool	是否保存到相册	N	true
dataType	string	结果数据格式： dataURL：base64 编码的图片数据 fileURL：图片在文件系统中的 URL（图片存放于临时目录中，退出时被清除） none：不返回数据（用于保存到相册的情况）	N	"none"
imageFormat	string	jpg / png	N	"jpg"
quality	int	jpg 的图片质量, 取值 1 到 100	N	75
maxWidth	int	图片的最大宽度. 过大将被等比缩小	N	-
maxHeight	int	图片的最大高度. 过大将被等比缩小	N	-
fn	function	回调函数	N	-

出参

回调函数带入的参数 result: {success, fileUrl, dataURL}。

名称	类型	描述
success	bool	是否处理成功
fileUrl	string	图片在文件系统中的 URL
dataURL	string	base64 编码的图片数据

错误码描述

错误码	描述
10	相册保存失败
11	图片文件保存失败

4.6.3 RPC 调用

重要：由于 JS 传入的 JSON 数据无法包含数据类型，在 Native 层转为字典时可能会由于数据类型问题导致误差，如果是数字类型的精确值，尽量使用字符串来进行传递。例如：{"value":9.45} 会被 native 转为 {"value":9.449999999999999} 然后上发到服务端。应该改为使用 {"value":"9.45"} 来传递。

RPC 接口的使用方法

```
AlipayJSBridge.call('rpc', {
  operationType: 'alipay.client.xxxx',
  requestData: [],
  headers: {}
}, function(result) {
  console.log(result);
});
```

代码示例

```
<h1>点击按钮发起 RPC 请求</h1>

<a href="javascript:void(0)"class="btn rpc">发起请求</a> <br/>
<a href="javascript:void(0)"class="btn rpcHeader">发起有响应头返回的请求</a>

<script>
function ready(callback) {
  // 如果 jsbridge 已经注入则直接调用
  if (window.AlipayJSBridge) {
    callback && callback();
  } else {
    // 如果没有注入则监听注入的事件
    document.addEventListener('AlipayJSBridgeReady', callback, false);
  }
}

ready(function() {
  document.querySelector('.rpc').addEventListener('click', function() {
    AlipayJSBridge.call('rpc', {
      operationType: 'alipay.client.xxxx',
      requestData: [],
      headers: {}
    }, function(result) {
      alert(JSON.stringify(result));
    });
  });
});

document.querySelector('.rpcHeader').addEventListener('click', function() {
  AlipayJSBridge.call('rpc', {
    operationType: 'alipay.client.xxxx',
    requestData: [],
    headers: {},
    getResponse: true
  }, function(result) {
    alert(JSON.stringify(result));
  });
});
});
</script>
```

API

```
AlipayJSBridge.call('rpc', {
  operationType,
  requestData,
  headers
}, fn);
```

入参

名称	类型	描述	必选	默认值
operationType	string	RPC 服务名称	Y	-
requestData	array	RPC 请求的参数。需要开发者根据具体 RPC 接口自行构造	N	-
headers	object	RPC 请求设置的 headers	N	{}
gateway	string	网关地址	N	alipay 网关
compress	boolean	是否支持 request gzip 压缩	N	true
disableLimitView	boolean	RPC 网关被限流时是否禁止自动弹出统一限流弹窗。	N	false
timeout	int	RPC 超时时间，单位为秒	N	框架统一设置，策略较复杂 iOS 端 Wi-Fi 环境 20s，其它环境 30s。 Android 端 Wi-Fi/4G 环境 12s 到 42s 之间。其它环境 32s 到 60s 之间
getResponse	boolean	获取 RPC 响应头（注意：如果设置为 true 的话，响应数据会多一层嵌套，可用于数据回流上报获取 traceId/entityId）	N	false
fn	function	回调函数	N	-

出参

回调函数带入的参数 result: {error }。

名称	类型	描述
error	string	错误码

错误码描述

错误码	描述
10	网络错误
11	请求超时
其他	由 mobilegw 网关定义

RPC 原生错误码

错误码	描述
1000	成功
0	未知错误
1	客户端找不到通讯对象
2	客户端没有网络 (JSAPI 做了转换, 返回 10)
3	客户端证书错误
4	客户端网络连接超时
5	客户端网络速度过慢
6	客户端请求服务端没返回
7	客户端网络 IO 错误
8	客户端网络请求调度错误
9	客户端处理错误
10	客户端数据反序列化错误, 服务端数据格式有误
11	客户端登录失败
12	客户端登录账号切换
13	请求中断错误, 例如线程中断时网络请求会被中断
14	客户端网络缓存错误
15	客户端网络授权错误
16	DNS 解析错误
17	operationType 不在白名单
1001	拒绝访问
1002	调用次数超过限制: 系统繁忙, 请稍后再试。
2000	登录超时, 请重新登录
3000	缺少操作类型或者此操作类型不支持
3001	请求数据为空: 系统繁忙, 请稍后再试
3002	数据格式有误
4001	服务请求超时, 请稍后再试
4002	远程调用业务系统异常: 网络繁忙, 请稍后再试
4003	创建远程调用代理失败: 网络繁忙, 请稍后再试
5000	未知错误: 抱歉, 暂时无法操作, 请稍后再试
6000	RPC-服务找不到
6001	RPC-目标方法找不到
6002	RPC-参数数目不正确
6003	RPC-目标方法不可访问
6004	RPC-JSON 解析异常
6005	RPC-调用目标方法时参数不合法
6666	RPC-业务异常

7000	没有设置公钥
7001	验签的参数不够
7002	验签失败
7003	验签时间戳校验失败
7004	验签 RPC 接口 operationType 参数为空
7005	productId 参数为空
7006	验签接口 did 参数为空
7007	验签接口 请求发送时间参数 t 为空
7008	验签接口 IMEI (客户端设备标识) 参数为空
7009	验签接口 IMSI (客户端用户标识) 为空
7010	验签接口 API 版本号为空
7011	验签接口 用户没有权限
7012	验签接口 RPC 没有对外开放
7013	验签接口 productId 没有注册或者获取密钥为空
7014	验签接口 加签数据为空
7015	验签接口 签约无效
7016	验签接口 请求登录 RPC 传入 sid 为空
7017	验签接口 请求登录 RPC 传入 sid 无效
7018	验签接口 请求登录 RPC 传入 token 无效
7019	验签接口 请求登录 RPC 获取 alipayuserid 为空
8001	etag : 响应数据没有变化

RPC 自定义 gateway

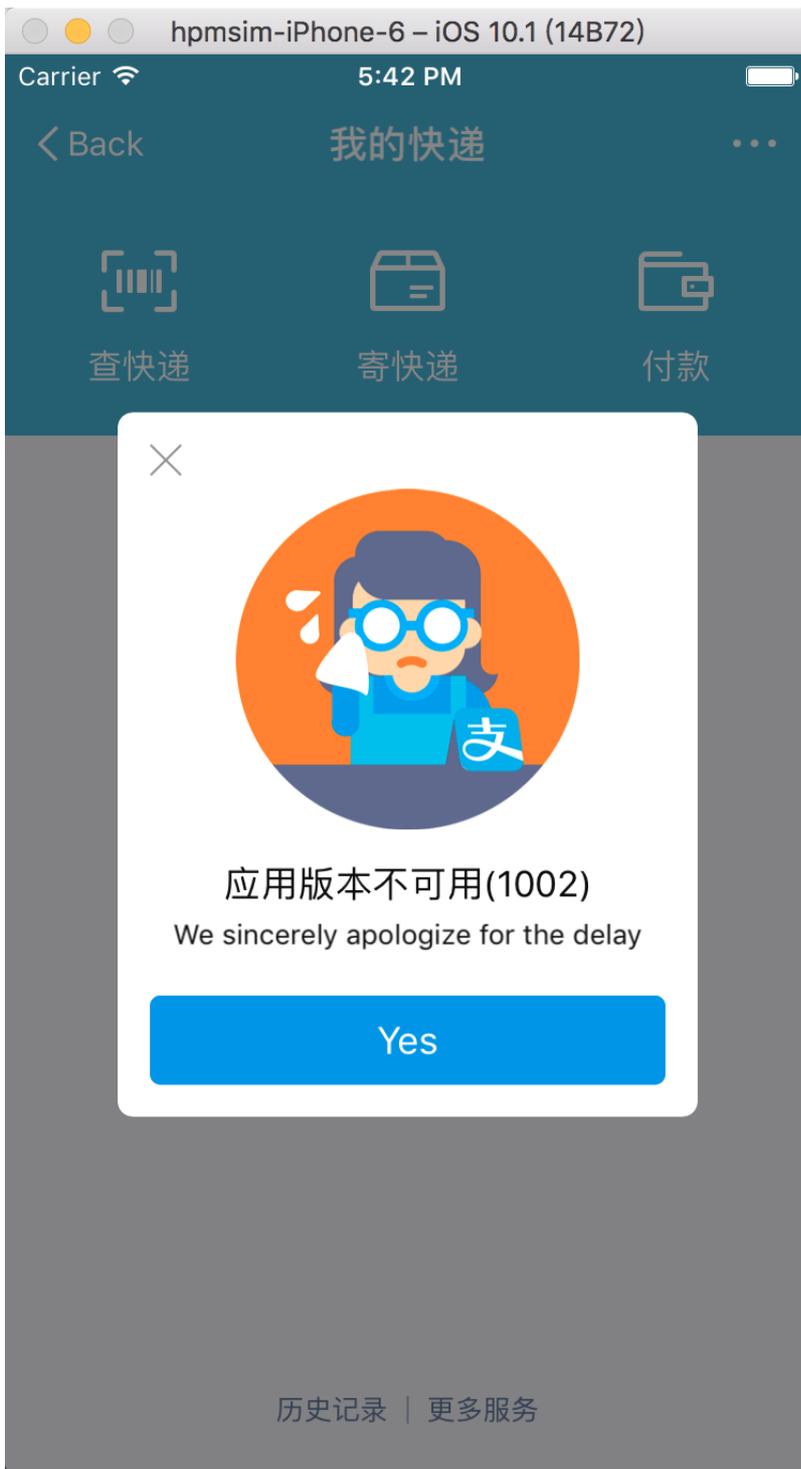
可在 RPC 调用中指定请求的网关地址。

RPC 限流逻辑

容器版本	disableLimitView	行为	回调参数
<=9.9.5	true	静默	1002
<=9.9.5	false	Alert	1002
>=9.9.6	true	静默	1002
>=9.9.6	false	网关处理	100201

行为类型	描述
静默	无
Alert	弹出统一限流框，如下图
Toast	弹出系统 Toast，如果用户关闭系统则没有
网关处理	根据网关的 RPC 配置，静默 Alert Toast

RPC 限流弹框



4.6.4 上报埋点

此接口为前端能用到的最原始的埋点接口。

remoteLog 接口的使用方法

```
AlipayJSBridge.call('remoteLog', {
```

```
bizType:"Nebula", // 业务类型
logLevel: 1, // 1 - high, 2 - medium, 3 - low
actionId:"event", // 埋点类型, 固定为"event"
seedId:"Login", // 埋点唯一标识
param1:"",
param2:"",
param3:"",
param4:"user=admin", // 自定义参数
});
```

说明：

- 如果您需要添加自定义埋点参数，可以通过 key=value 的格式添加至上述代码的 param4 中，如：
: param4:"user=admin"。
- 添加多个自定义埋点参数时，在 param4 中添加的内容格式如下：
: param4:"key1=value1,key2=value2,key3=value3"。

代码示例

```
<h1>点击按钮会记录相关信息</h1>

<a href="javascript:void(0)"class="btn read">点一点</a>

<script>
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}

ready(function() {
document.querySelector('.read').addEventListener('click', function() {
AlipayJSBridge.call('remoteLog', {
type:"behavior",
bizType:"Nebula",
spmId:"a1.b2",
logLevel: 1, // 1 - high, 2 - medium, 3 - low
actionId:"event"
seedId:"xxx",
param1:"xxx",
param2:"xxx",
param3:"xxx",
param4:"xxx",
});
});
});
});
</script>
```

API

```
AlipayJSBridge.call('remoteLog', {
  type, seedId, ucid, bizType, logLevel,
  actionId, spmId, param1, param2, param3, param4
});
```

入参

名称	类型	描述	必选	默认值
type	string	埋点类型：monitor（监控类型）、monitorWithLocation（监控类型，自动在 param4 里带上经纬度）、behavior（行为类型）、behaviorAuto（自动行为类型）、performance（性能类型）、error（异常类型，9.6.8 版本开始支持）、135（135 业务相关，9.9 版本开始支持）	N	"monitor"
seedId	string	埋点 ID	Y	""
bizType	float	业务类型标识，该参数传值时，会生成单独的日志文件	N	-
logLevel	int	1 - high, 2 - medium, 3 - low, 低级别的日志可能会被限流	N	-
actionId	string	埋点类型，固定为 event	N	""
spmId	string	spm 编码。当编有 spmId 时，忽略 seedId	Y	""
param1	string	埋点参数 1	N	""
param2	string	埋点参数 2	N	""
param3	string	埋点参数 3	N	""
param4	string	埋点参数 4	N	""

4.6.5 设置 AP 数据

- 实现 H5LoginProvider 接口 Android 代码示例设置 H5LoginProvider
- setAPDataStorage 接口的使用方法 代码示例API入参出参错误码描述

setAPDataStorage 接口用于保存一个字符串到客户端统一存储，字符串长度不得超过 200×1024。

说明：

- 底层存储服务组件在 iOS 和 Android 中实现不一致。Android 统一存储组件不支持 type=user 属性，为了与前端接口一致，当设置 type=user 时，Android 底层会设置为 key=key + "_" + MD5(userId + userId + userId) 并进行存储。业务用客户端取的时候也要对 key 做相应处理。
- 在 10.1.60 及以下版本的基线中，客户端需要做适配工作才能使接口能够获得 userId，否则存储接口将无法按 userId 区分存储，参见下方 实现 H5LoginProvider 接口。
- 在 10.1.68 及以上版本的基线中，userId 默认使用 MILogger.setUserId 中的值，若实现 H5LoginProvider，则取用 H5LoginProvider。

实现 H5LoginProvider 接口

Android

实现 H5LoginProvider 接口，并将实例类设置到 H5ProviderManager 中。

代码示例

```
package com.mpaas.nebula.provider;

import android.os.Bundle;

import com.alipay.mobile.common.logging.api.LoggerFactory;
import com.alipay.mobile.nebula.provider.H5LoginProvider;

public class H5LoginProviderImpl implements H5LoginProvider {
    // 其他代码省略

    @Override
    public String getUserId() {
        // 此方法返回 userId 即可
        return LoggerFactory.getLogContext().getUserId();
    }

    // 其他代码省略
}
```

设置 H5LoginProvider

```
H5Utils.setProvider(H5LoginProvider.class.getName(), new H5LoginProviderImpl());
```

setAPDataStorage 接口的使用方法

```
AlipayJSBridge.call('setAPDataStorage', {
    type:"common",
    business:"customBusinessKey",
    key:"customKey",
    value:"customValue"
}, function(result) {
    alert(JSON.stringify(result));
});
```

```
});
```

代码示例

```
<button id="J_saveDataBtn" class="btn">保存数据</button>
<button id="J_getDataBtn" class="btn">查看数据</button>
<button id="J_removeDataBtn" class="btn">删除数据</button>

<script>
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}

ready(function() {
document.querySelector('#J_saveDataBtn').addEventListener('click', function(e) {
AlipayJSBridge.call('setAPDataStorage', {
type:"common",
business:"customBusinessKey",
key:"customKey",
value:"customValue"
}, function(result) {
alert(JSON.stringify(result));
});
}, false);

document.querySelector('#J_getDataBtn').addEventListener('click', function(e) {
AlipayJSBridge.call('getAPDataStorage', {
type:"common",
business:"customBusinessKey",
key:"customKey"
}, function(result) {
alert(JSON.stringify(result));
});
}, false);

document.querySelector('#J_removeDataBtn').addEventListener('click', function(e) {
AlipayJSBridge.call('removeAPDataStorage', {
type:"common",
business:"customBusinessKey",
key:"customKey"
}, function(result) {
alert(JSON.stringify(result));
});
}, false);
}, false);
</script>
```

API

```
AlipayJSBridge.call('setAPDataStorage', {
  type, business, key, value
});
```

入参

名称	类型	描述	必选	默认值
type	string	(user/common) 用户维度存储还是公共存储，默认值 common	N	"common"
business	string	自定义的业务标识，可与相应的客户端存取代码约定，默认值为：NebulaBiz。 (在 Android 中，该业务标识与创建 APSharedPreferences 时所传入的 GROUP_ID 对应)	N	""
key	string	自定义数据的 key	Y	""
value	string	需要存储的值，仅支持字符串类型。JSON 数据需要先字符串化	Y	""

出参

回调函数带入的参数 result: {success}。

名称	类型	描述
success	bool	是否保存成功

错误码描述

错误码	描述
11	字符串长度超限

4.6.6 获取 AP 数据

此接口用于从统一存储中获取数据，仅支持字符串类型。

getAPDataStorage 接口的使用方法

```
AlipayJSBridge.call('getAPDataStorage', {
  type:"common",
  business:"customBusinessKey",
  key:"customKey",
}, function(result) {
  alert(JSON.stringify(result));
});
```

代码示例

```
<button id="J_saveDataBtn" class="btn">保存数据</button>
<button id="J_getDataBtn" class="btn">查看数据</button>
<button id="J_removeDataBtn" class="btn">删除数据</button>

<script>
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
if (window.AlipayJSBridge) {
callback && callback();
} else {
// 如果没有注入则监听注入的事件
document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}

ready(function() {
document.querySelector('#J_saveDataBtn').addEventListener('click', function(e) {
AlipayJSBridge.call('setAPDataStorage', {
type:"common",
business:"customBusinessKey",
key:"customKey",
value:"customValue"
}, function(result) {
alert(JSON.stringify(result));
});
}, false);

document.querySelector('#J_getDataBtn').addEventListener('click', function(e) {
AlipayJSBridge.call('getAPDataStorage', {
type:"common",
business:"customBusinessKey",
key:"customKey"
}, function(result) {
alert(JSON.stringify(result));
});
}, false);

document.querySelector('#J_removeDataBtn').addEventListener('click', function(e) {
AlipayJSBridge.call('removeAPDataStorage', {
type:"common",
business:"customBusinessKey",
key:"customKey"
}, function(result) {
alert(JSON.stringify(result));
});
}, false);
}, false);
</script>
```

API

```
AlipayJSBridge.call('getAPDataStorage', {
type, business, key
});
```

入参

名称	类型	描述	必选	默认值
type	string	(user/common) 用户维度存储还是公共存储，默认值为：common	N	"common"
business	string	自定义的业务标识，可与相应的客户端存取代码约定，默认值为：NebulaBiz。 (在 Android 中，该业务标识与创建 APSharedPreferences 时所传入的 GROUP_ID 对应)	N	""
key	string	自定义数据的 key	Y	""

出参

回调函数带入的参数 result: {data}。

名称	类型	描述
data	string	数据
errorMessage	string	未找到该数据

错误码描述

错误码	描述
11	未找到该数据

4.6.7 移除 AP 数据

此接口用于从统一存储中删除数据。

removeAPDataStorage 接口的使用方法

```
AlipayJSBridge.call('removeAPDataStorage', {
  type:"common",
  business:"customBusinessKey",
  key:"customKey",
}, function(result) {
  alert(JSON.stringify(result));
});
```

代码示例

```
<button id="J_saveDataBtn" class="btn">保存数据 </button>
<button id="J_getDataBtn" class="btn">查看数据 </button>
<button id="J_removeDataBtn" class="btn">删除数据 </button>

<script>
function ready(callback) {
// 如果 jsbridge 已经注入则直接调用
```

```

if (window.AlipayJSBridge) {
  callback && callback();
} else {
  // 如果没有注入则监听注入的事件
  document.addEventListener('AlipayJSBridgeReady', callback, false);
}
}

ready(function() {
  document.querySelector('#J_saveDataBtn').addEventListener('click', function(e) {
    AlipayJSBridge.call('setAPDataStorage', {
      type:"common",
      business:"customBusinessKey",
      key:"customKey",
      value:"customValue"
    }, function(result) {
      alert(JSON.stringify(result));
    });
  }, false);

  document.querySelector('#J_getDataBtn').addEventListener('click', function(e) {
    AlipayJSBridge.call('getAPDataStorage', {
      type:"common",
      business:"customBusinessKey",
      key:"customKey"
    }, function(result) {
      alert(JSON.stringify(result));
    });
  }, false);

  document.querySelector('#J_removeDataBtn').addEventListener('click', function(e) {
    AlipayJSBridge.call('removeAPDataStorage', {
      type:"common",
      business:"customBusinessKey",
      key:"customKey"
    }, function(result) {
      alert(JSON.stringify(result));
    });
  }, false);
}, false);
</script>

```

API

```

AlipayJSBridge.call('removeAPDataStorage', {
  type, business, key
});

```

入参

名称	类型	描述	必选	默认值
type	string	(user/common) 用户维度存储还是公共存储，默认值为：common	N	"common"
busin	string	自定义的业务标识，可与相应的客户端存取代码约定，默认值为：NebulaBiz。	N	""

ess	ng	(在 Android 中, 该业务标识与创建 APSharedPreferences 时所传入的 GROUND_ID 对应)		
key	string	自定义数据的 key	Y	""

出参

回调函数带入的参数 result: {success}。

名称	类型	描述
success	bool	删除是否成功

