WebSDK接口文档

统一的交互说明

- 不支持界面：不支持H5时，会拦截加载，并显示默认的功能不支持页面。
- 加载失败界面：在页面加载失败时，统一显示加载错误页面，而不会出现网页404、502等错误页面。
- 加载动画界面：将加载动画界面抽象为接口，使用方实现加载动画接口，并设置到WebManager中，在网页加载过程中会显示。
- 右滑 H5 返回上一页：右滑默认会退出当前Activity ，但是在H5中需要返回上一级页面，所以增加了该处理。

支持的Js接口

1. 退出当前页面

native方法名：**commonApi.closePage**

2. 判断某个方法是否支持

Native会提供多个方法给Js，但是有些方法存在版本兼容性问题，需要Js判断某个版本是否支持某方法。native方法名：**commonApi.hasNativeMethod**

参数	必须	类型	说明
name	是	string	方法名
type	是	string	方法类型：同步sync,异步async，所有all

```
//返回值: boolean
function checkMethod(){
    var result = dsBridge.call("commonApi.hasNativeMethod", {name: "getAp
pData", type:"async"});
}
```

3. 启动应用接口

Js传入序列化的Intent，可以启动任意应用指定页面。
native方法名：**openURL**

参数名	必须	类型	说明
packageName	是	string	应用包名
uri	是	string	打开具体应用的Intent 序列化后
flag	是	string	序列化flag,1或者2

返回数据类型 : json

状态码	说明
0	开启成功
1	应用不存在，需要安装应用
2	页面不存在，需要更新应用
3	功能不显示
4	功能被禁用

```
{
    "code": "0",
    "desc": "success",
}
```

4. SharedPreferences读写接口：Js持久化数据到Native。

5. Toast接口

Js调用Native的Toast方法。native方法名：**toastMethod**

参数名	必须	类型	说明
msg	是	string	提示语

6. 本地图片读接口

Js可以传入Base64编码的字符串，Native将其解析为图片文件并存储到指定目录，读取反之。
native方法名：**xtc.getLocalImage**

参数名	必须	类型	说明
path	是	string	图片路径
width	否	int	图片宽度，会进行压缩处理，没有则使用屏幕宽度

height	否	int	图片高度，会进行压缩处理，没有则使用屏幕宽度

返回数据类型：json

状态码	说明
000001	获取成功
000002	文件不存在

```
{
    "code": "000001",
    "image": "图片的base64",
}
```

7. 保存图片至本地

native方法名：**xtc.saveLocalImage**

参数名	必须	类型	说明
path	是	string	图片路径
image	是	string	图片的Base64
name	是	string	图片名称，包含后缀

返回数据类型：boolan

8. 应用信息查询接口

Js可以传入packageName，查询该应用当前的版本号、版本名称。native方法名：**xtc.getAppInfo**

- **调用参数说明：应用包名（string）**
- **返回数据类型：json**

状态码	说明
000001	获取成功
000002	应用不存在

```
{
    "code": "000001",
    "appInfo": {
        "packageName":"com.xtc.setting",
```

```
            "appName":"设置",
            "versionCode":30000,
            "versionName":"3.0.0"
        }
    }
```

9. 生命周期回调

分享接口、H5等需要关注页面声明周期的调用情况，所以增加了一个分发逻辑。

10. 分享接口

Js支持调用分享接口，Native 需要同时引入WebSdk和ShareSdk。

10.1 通用参数

- 分享场景 `ReqShareScene`

参数名	必须	类型	说明
type	是	int	分享场景
chat	否	Chat	选择微聊场景时需要，如果没有采用默认
moment	否	Moment	选择好友圈场景时需要，如果没有采用默认

- 分享场景Type（场景和类型求与，例如：0x10 & 1 为静默分享至微聊）

```
//微聊
int TYPE_CHAT = 1;
//好友圈
int TYPE_MOMENT = 2;

//静默分享
int SILENT = 0x10;
//标准分享
int STANDARD = 0x20;
```

- 微聊场景 `Chat`

参数名	必须	类型	说明
friendType	否	int	分享的好友类型，默认为好友与好友圈
filterConversationList	否	List	过滤的会话Id
filterTip	否	List	过滤的会话Id点击后的提示语
filterModeList	否	List	过滤的机型

- 好友类型 `friendType`（类型相加时求或，例如好友与好友群：0x00000001 | 0x00000010）

```
//好友
int FRIEND = 0x00000001;
//好友群
int FRIEND_GROUP = 0x00000010;
//管理员
int MANAGER = 0x00000100;
//家庭圈
int FAMILY_GROUP = 0x00001000;
```

- 好友圈场景 `Moment`（暂无可选择参数）

```
//accountType取值
手表账户 = 1
手机账户 = 0
家庭群聊 = -1

//account取值
好友：watchId
家庭群聊：null
```

10.2 通用响应

状态码	说明
-1	没有分享
1	分享成功
2	取消分享
3	鉴权失败
4	当天分享次数超限
5	类型不支持
6	参数错误
7	会话列表拉取失 败
8	网络异常
9	token异常
10	该场景已被禁用
11	无可分享的好友

100	未知异常
1000	点击跳转状态码

```
{
    "code": 1,
    "errorDesc": "send share success !",
    "transaction": "1564038823094dab3dc3b-995c-4524-bca3-e2263674da73"
    "conversationId": "nZj+mIJVxRDYriVVuCgnOg==", // 判断分享同个好友的唯一标识
}
```

10.3 分享文本

- native方法名：**xtc.shareText**
- **调用参数说明：**

参数名	必须	类型	说明
key	是	string	每个App唯一，找分享库维护人员申请
content	是	string	文本内容
appName	否	string	应用名称
icon	否	string	应用图标
scene	否	ReqShareScene	分享的场景，为空时弹出场景选择界面

10.4 分享图片

- native方法名：**xtc.shareImage**
- **调用参数说明：**

参数名	必须	类型	说明
key	是	string	每个App唯一，找分享库维护人员申请
image	是	string	图片的Base64
appName	否	string	应用名称
icon	否	string	应用图标
scene	否	ReqShareScene	分享的场景，为空时弹出场景选择界面

10.5 分享程序

- native方法名：**xtc.shareApp**
- **调用参数说明：**

参数名	必须	类型	说明
key	是	string	每个App唯一，找分享库维护人员申请
image	是	string	图片的Base64
desc	是	string	内容概要描述信息
extInfo	否	string	扩展信息，点击启动应用时原样传递
startActivity	否	string	需要启动的页面全类名，如果没有则启动分享的页面
appName	否	string	应用名称
icon	否	string	应用图标
scene	否	ReqShareScene	分享的场景，为空时弹出场景选择界面

性能优化

- 本地拦截 vue.js 加载策略：vue.js属于通用框架，与业务无关，所以将该文件放在本地assert目录，拦截网络加载，直接加载本地文件，可以提高加载速度。

基本使用

```
//rlWebView必须为RelativeLayout，其作为动画界面、H5界面、加载失败界面等的父布局
WebManager webManager = WebManager.getInstance(this,rlWebView);

//设置加载动画界面，CustomerLoading为自定义的加载动画，其需要实现LoadingView抽象类
webManager.setLoadingView(new CustomerLoading(this));

//开始加载
webManager.loadUrl("http://www.baidu.com");

//设置加载失败接口
webManager.setLoadFailListener(new WebManager.LoadFailListener() {
    @Override
    public void onLoadFail(String url) {

    }
});

//设置加载成功接口
webManager.setLoadSuccessListener(new WebManager.LoadSuccessListener() {
    @Override
    public void onLoadSuccess(String url) {
```

```
        }
    });
```

生命周期分发

H5的多个接口可能用到，尽量都实现，否则可能导致H5接口调用异常。
```

```
Override
protected void onCreate(Bundle savedInstanceState) {
LifecycleDispatcher.getInstance().disPatchOnCreate(savedInstanceState);
}

Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
super.onActivityResult(requestCode, resultCode, data);
LifecycleDispatcher.getInstance().dispatchActivityResult(requestCode,resultCode,data);
}

Override
protected void onStart() {
super.onStart();
LifecycleDispatcher.getInstance().dispatchOnStart();
}

Override
protected void onResume() {
super.onResume();
LifecycleDispatcher.getInstance().disPatchOnResume();
}

Override
protected void onNewIntent(Intent intent) {
super.onNewIntent(intent);
LifecycleDispatcher.getInstance().dispatchOnNewIntent(intent);
}

Override
protected void onStop() {
super.onStop();
LifecycleDispatcher.getInstance().dispatchOnStop();
}

Override
protected void onDestroy() {
```
```

```
        super.onDestroy();
        LifecycleDispatcher.getInstance().dispatchOnDestroy();


            //一定要释放
            if (webManager!= null) {
                webManager.release();
            }


        }
```