2017-12-01Dcloud打包常见问题

Dcloud 打包 APP 常见问题

1 . Android 手机的 back 键直接退出应用的问题

```
<script>
    document.addEventListener('plusready', function() {
            var webview = plus.webview.currentWebview();
            plus.key.addEventListener('backbutton', function() {
                webview.canBack(function(e) {
                    if(e.canBack) {
                        webview.back();
                    } else {
                        //webview.close(); //hide,quit
                        //plus.runtime.quit();
                        mui.plusReady(function() {
                            //首页返回键处理
                            //处理逻辑：1秒内，连续两次按返回键，则退出应用；
                            var first = null;
                            plus.key.addEventListener('backbutton', function() {
                                //首次按键，提示‘再按一次退出应用’
                                if (!first) {
                                    first = new Date().getTime();
                                    mui.toast('再按一次退出应用');
                                    setTimeout(function() {
                                        first = null;
                                    }, 1000);
                                } else {
                                    if (new Date().getTime() - first < 1500) {
                                        plus.runtime.quit();
                                    }
                                }
                            }, false);
                        });
                    }
                })
            });
        });
    </script>
```

2 . 个推iOS 和 Android 端的消息处理 （Dcloud 推送开发指南 http://ask.dcloud.net.cn/article/34）

最近做的app中使用到了推送，iOS 和 Android 的处理方式不一样，Android 只需要把平台生成的 appid，appkey，appsecr

et 添加到 sdk 配置里面就可以直接推送消息， 而 iOS 只支持传透消息，iOS 还区分应用在线和不在线，两个系统的处理方式完全不同；

1 . Android : 应用在前台会直接受到传透消息，然后自己去处理；

如果后台不杀死push进程收到透传我去创建本地推送在通知栏显示plus.push.createMessage(str, jsonData, options);杀死push进程什么也收不到了。当再次打开应用会收到之前的推送。(问了官方说android杀死应用就是收不到郁闷)；

2 . iOS : 应用在前台会直接收到透传然后自己去处理；

应用关闭和在后台就需要将消息推送个APNS然后APNS在推送给ios手机。(如果后台和关闭状态收不到推送就有可能是 ios的正式配置有问题 你需要一个ios的哥们帮你了)。

3 . android的push监听可以写很多都没事而ios你多写了它会多次调用push监听方法。

```
var Push = function() {


}
/**
 *
 * 推送的监听
 */
Push.prototype.pushListener = function () {
    var pointer = this;
    plus.push.addEventListener("click", function (msg) {
        switch (msg.payload) {
            case "LocalMSG":
                ApiConfig.staticToast("点击本地创建消息启动: ");
                break;
            default:
                ApiConfig.staticToast("点击离线推送消息启动: ");
                break;
        }
        if (msg.payload) {
            pointer.handle(msg);
        }
    }, false);
    plus.push.addEventListener("receive", function (msg) {
        if (msg.aps) { // Apple APNS message
            ApiConfig.staticToast("接收到在线APNS消息: ");
        } else {
            ApiConfig.staticToast("接收到在线透传消息: ");
        }
        if (plus.os.name == 'iOS') {
            if (msg.payload) {
                pointer.notificationMessage(msg);
            }
        } else {
            pointer.notificationMessage(msg);
        }
    }, false);
```

```
}

/**
 * 根据推送消息在通知栏中显现对应的提示
 * @param {Object} msg
 */
Push.prototype.notificationMessage = function(msg) {
    ApiConfig.staticToast(msg, 1);
    ApiConfig.staticIsDebug("notificationMessage", msg, 1);
    var content = '';//你要展示的提示
    var pointer = this;
    var jsonData = '';
    switch(plus.os.name) {
        case "Android":
            jsonData = eval("(" + msg.payload + ")");
            break;
        case "iOS":
            jsonData = msg.payload;
            break;
    }
    pointer.createLocalPushMsg(msg, content);
}

/**
 *  处理透传消息
* @param {Object} msg
* @param {Object} content
 */
Push.prototype.createLocalPushMsg = function(msg, content) {
    //创建一个符合你自己要显示推送通知
        pointer.createMessage(str, jsonData, options);
}

/**
 *  创建本地推送消息
 * @param {Object} str
 * @param {Object} jsonData
 * @param {Object} options
 */
Push.prototype.createMessage = function(str, jsonData, options) {
    switch(plus.os.name) {
        case "Android":
            jsonData = jsonData;
            break;
        case "iOS":
```

```
                jsonData = jsonData.eid;
                break;
        }
        plus.push.createMessage(str, jsonData, options);
    }

    /**
     * 处理通知方法
     * @param {Object} msg
     */
    Push.prototype.handle = function(msg) {
            //清除ios推送小红点
        this.cancelPushclear();
    }
    Push.prototype.cancelPushClear = function () {
      plus.push.clear();
    }
```

3 . Dcloud 检测服务器版本升级

这里的 check 服务端的 update.json 文件的时候不要用 mui 提供的ajax操作;

```
var server = lib.UpdateSrc, //获取升级描述文件服务器地址
      curVersion = null; //APP当前的版本号
      /**
       * 检查升级数据
       */
getUpdateData();
function checkUpdateData(j) {
    var inf = j[plus.os.name];
    var srvVer = inf.version;
    curVersion = plus.runtime.version;
    //curVersion = '1.0.2';
    // 提示用户是否升级
    if(compareVersion(curVersion, srvVer)) {
        plus.nativeUI.confirm(inf.note, function(i) {
            if(0 == i.index) {
                plus.runtime.openURL(inf.url);
            } else if(1 == i.index && inf.isforce) {
                mui.alert('升级新版本后体验更佳哦!');
                setTimeout(function() {
                    plus.runtime.quit();
                }, 1500)
            } else {
                mui.alert('升级版本后体验更佳哦!');
            }
```

```
        }, inf.title, ["立即更新", "跳过此版本"]);
    }
}
/**
 * 从服务器获取升级数据
 */
function getUpdateData() {
    $.getJSON(server, function(data) {
        console.log(data);
        checkUpdateData(data);
    })
}
/**
 * 比较版本大小，如果新版本nv大于旧版本ov则返回true，否则返回false
 * @param {String} ov
 * @param {String} nv
 * @return {Boolean}
 */
 function compareVersion(ov, nv) {
    if(!ov || !nv || ov == "" || nv == "") {
            return false;
    }
    var b = false,
          ova = ov.split(".", 4),
          nva = nv.split(".", 4);
    for(var i = 0; i < ova.length && i < nva.length; i++) {
        var so = ova[i],
              no = parseInt(so),
              sn = nva[i],
              nn = parseInt(sn);
        if(nn > no || sn.length > so.length) {
            return true;
        } else if(nn < no) {
            return false;
        }
    }
    if(nva.length > ova.length && 0 == nv.indexOf(ov)) {
        return true;
    }
 }
```

4 . 消息推送

iOS 和 Android 端对推送的处理方式不一样

首先，消息推送分为普通消息和消息透传；

1 . 普通消息：暂只支持Android平台；

2 . 透传消息：iOS 应用在线或者是离线都可以收到透传消息；

Android 只有在应用在线或者是后台运行的时候可以收到消息，杀掉应用进程是无法收到消息；

需要注意的是：消息推送功能，用户在首次下载APP的时候，会提示一个是否接收应用推送的消息之类的消息提示，如果这里用户没有允许的话，后面的推送消息都无法接收到；