WEEX Conf

Weex在盛大游戏中的应用实践

李永亮（kissy小鬼）

说说下面5个话题

1 技术选型

2 小试牛刀

3 大规模应用

4 性能优化

5 踩坑填坑

技术选型

三端统一是公司大前端分会倡导下，自上而下的期许

WEEX

选中weex的理由

一、提升开发效率、交付速度

- 一次编写，多端运行
- 运用Web技术
- Vue语法简单，学习成本低
- 易复用已有Vue组件与APP组件功能

选中weex的理由

二、高性能、优秀的用户体验

- bundle体积小，下载快
- 渲染速度快，可做出原生的体验
- 对加载时间和资源占用深度优化

选中weex的理由

三、由阿里主持推进，N多APP在使用，可放一百个心

极客时间、众安保险、携程汽车票、饿了么、杭州尚妆、平安付、爱奇艺、百度、火猫直播、润和、上海连游、陆金所、和诚智汇、钱升钱、Movin、神州数码、第一财经新媒体、平安壹钱包、南方航空、一起作业、猎豹、NLE跨境物流、易知科技、来去网络、讯联、点我达、微一案、汽车之家、纽诺、执楠、美团点评、于斯课堂、聚美、东方财富网、惠普、国金、环球易购、掌门1对1、微软、加减法、亿德力、喜马拉雅、优酷、中赢、贝贝、浙江农商行、普悦软件、人人视频、中科富创看见音乐、电信、顶新集团、亿阁科技、美的集团、爱屋及屋、盛大游戏 …

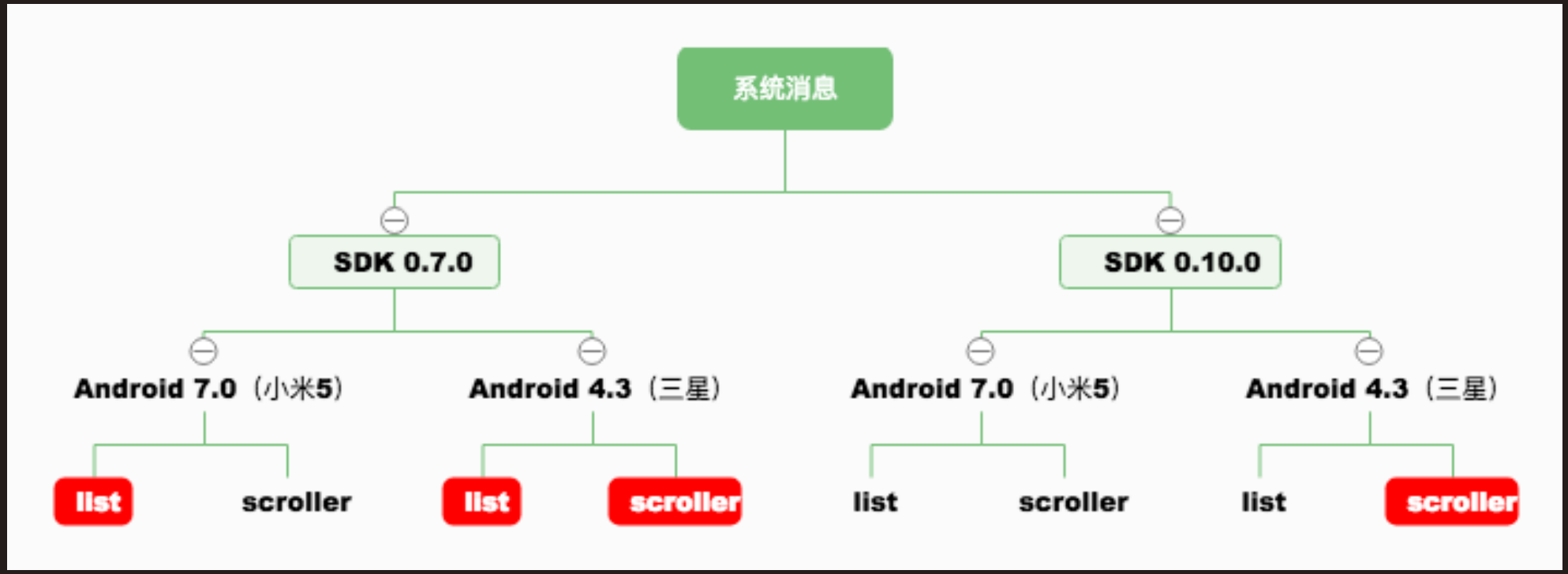
小试牛刀

系统消息、公告通知、食客在线三个页面

小试牛刀

小经验：尽量采用高版本的WEEX SDK，我们目前采用的是0.12.0

Weex sdk 0.7.0升级到0.10.0后列表的处理更优



小试牛刀

小经验：Weex页接口拥有登录态（Weex目前无cookie机制）

小试牛刀

借用Native Cookie的能力，是目前我们采用的方式

```
WX_EXPORT_METHOD(@selector(get:callback:))
- (void)get:(NSString *)url callback:(WXModuleCallback)callback {
    if ([GKitPrivacy httpsOnly]) {
        url = [url replaceAll:@"http://" with:@"https://"];
    }
    NSURLSessionDataTask *task  = [[NSURLSession sharedSession] dataTaskWithURL:[NSURL URLWithString:url] completionHandler:^(NSData * _Nullable data,
        NSURLResponse * _Nullable response, NSError * _Nullable error) {
        dispatch_async(dispatch_get_main_queue(), ^{
            if (!data) {
                callback(nil);
            }else{
                callback([[NSString alloc] initWithData:data encoding:NSUTF8StringEncoding]);
            }
        });
    }];
    [task resume];
}

WX_EXPORT_METHOD(@selector(login:))
- (void)login:(WXModuleCallback)callback{
    [GMMLogin login:^(NSInteger resultCode, NSString* resultMsg, NSString *ticket) {
        callback(@{@"return_code":@(resultCode),@"return_message":resultMsg});
    }];
}
```

小试牛刀

小经验：Weex优雅地跳转到Native并传参

采取封装module方法：gmmmodule.load(url, cb)

Weex访问的url形式如下：

sdggmm://product_detail?book_id=123&goods_type=456&game_id=789

cb参数在有些场景中可以使用（如）

```
// 评价
evaluate (action, title) {
    let vm = this
    utils.load(sdggmm.order_evaluate(vm.myDetail.order_id, vm.isBuyer ? '0' : '1'), (ret) => {
        if (ret.return_code == 0) {
            utils.sendEvent('tradeDetail', {})
        }
    })
},
```

小试牛刀

小经验：.we改造为vue版本

weex-toolkit在1.0.1之后才支持vue格式（SDK版本 >= 0.10.0）

- 建议直接采用vue版本来开发
- 借助weex-vue-migration工具实现语法转换
- 再手动调整一番就OK

大规模应用

在游戏代练业务中21个页面采用weex

在公司会议系统中7个页面采用Weex

大规模应用

首先需要解决的问题有很多

- 采用单页面还是多页?
- 页面跳转存在什么问题?
- 如何解决数据通信问题?
- 样式如何做到复用?
- 怎么充分利用Native?

WEEX

采用单页面还是多页?

一度在页面开发方式上迷茫，尝试了流行的SPA方式

SPA の优点

可避免重复加载资源，极少个bundle

可自定义专场效果

可立即展现，无需等待

可采用全局数据状态共享

采用单页面还是多页?

SPA固然有优点，也一定程度上影响产品体验和团队开发效率

SPA の缺点

首次打开的页面开销

内存管理不当，容易造成APP Crash

若干原生入口，面临复杂度

所有页面基于相同的框架，不具有灵活性

全家桶(vuex+vue-router...)学习成本高

团队协作差

采用单页面还是多页?

最终我们选择采用多页开发方式，有以下利好：

- 每个页面一个bundle
- 与原生保持一致的专场效果
- 充分实践运行时优化、缓存和预加载
- 不同团队/个人可自由选择JS框架
- 更好的内存管理，减少Crash的次数

页面跳转存在什么问题?

页面间的跳转遇到了问题...

我们期望这样子

结果却是这样子

页面跳转存在什么问题?

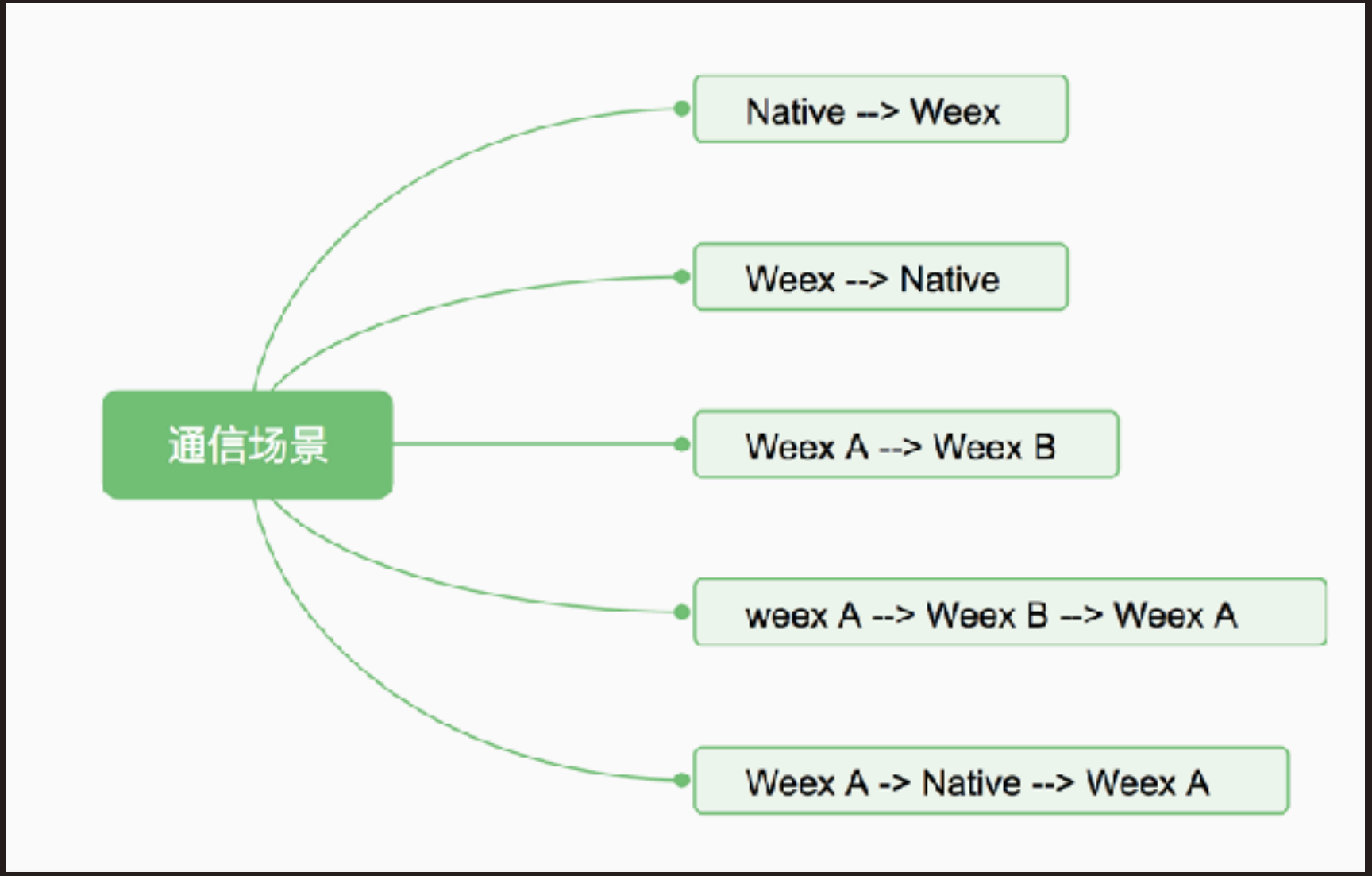
怎么办? 重写SDK中navigator的push方法

```
WX_EXPORT_METHOD(@selector(push:callback:))
- (void)push:(NSDictionary *)param callback:(WXModuleCallback)callback
{
    UIViewController *container = self.weexInstance.viewController;
    WXNavigationResultBlock block = ^(NSString *code, NSDictionary *responseData) {
        if (callback && code) {
            callback(code);
        }
    };

    if (0 == [param count] || !param[@"url"] || !container) {
        [self callback:block code:MSG_PARAM_ERR data:nil];
        return;
    }

    BOOL animated = YES;
    NSString *obj = [[param objectForKey:@"animated"] lowercaseString];
    if (obj && [obj isEqualToString:@"false"]) {
        animated = NO;
    }

    WXBaseViewController *vc = [[WXBaseViewController alloc]initWithSourceURL:[NSURL URLWithString:param[@"url"]]];
    vc.hidesBottomBarWhenPushed = YES;
    [container.navigationController pushViewController:vc animated:animated];
    container.navigationController.navigationBarHidden = YES;
    [self callback:block code:MSG_SUCCESS data:nil];
}
```

WEEX

如何解决数据通信问题?

全面数据通信方案，数据传递的场景还是很多的



如何解决数据通信问题?

Native -> Weex：原生页跳转到Weex页

Native在weex
实例变量config
上设置数据order_id

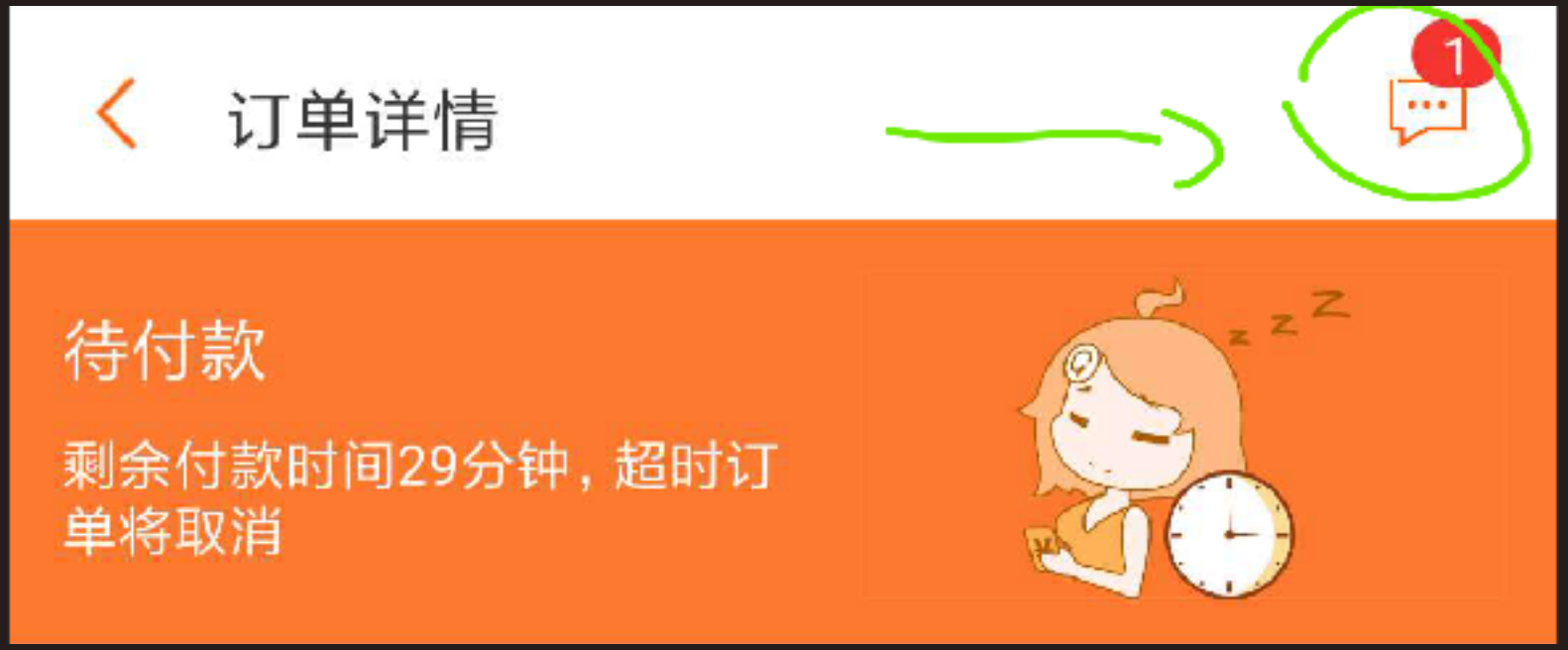
在weex页面中
vm.$getConfig().order_id
可以是复杂数据类型

{"bundleUrl":"/storage/emulated/0/Android/data/com.snda.mhh/cache/download/mhh_weex/dailian/order/detail/app.js","order_id":"DLTFW1000010000149934077980204 50","env":{"platform":"android","osVersion":"7.0","appVersion":"2.6.0","weexVersion":"0.11.0","deviceModel":"MI 5","appName":"com.snda.mhh","deviceWidth":"1080","deviceHeight":"1920","scale":"3.0"}}

OK

如何解决数据通信问题?

Native -> Weex：Native推送消息



如何解决数据通信问题?

Native -> Weex：Native推送消息

- Native订阅消息接收到消息发送事件给Weex页

```
@Subscribe(threadMode = ThreadMode.MAIN, sticky = true)
public void onChatUnreadCountEvent(ChatUnreadCountEvent event) {
    unreadCount = event.unreadCount;
    if(unreadCount > 0) {
        sendMsgEvent(unreadCount);
    }
}

private void sendMsgEvent(int unreadCount){
    List<WXSDKInstance> instances = WXSDKManager.getInstance().getWXRenderManager().getAllInstances();
    Map<String,Object> map = new HashMap<>();
    map.put("unreadCount", unreadCount);
    for(WXSDKInstance instance : instances) {
        instance.fireGlobalEventCallback("msgEvent", map);
    }
}
```

- Weex监听并接收数据

```
globalEvent.addEventListener('msgEvent', function (data) {
    console.log(data)
    vm.name = data.unreadCount || 0
})
```

如何解决数据通信问题?

Weex -> Native

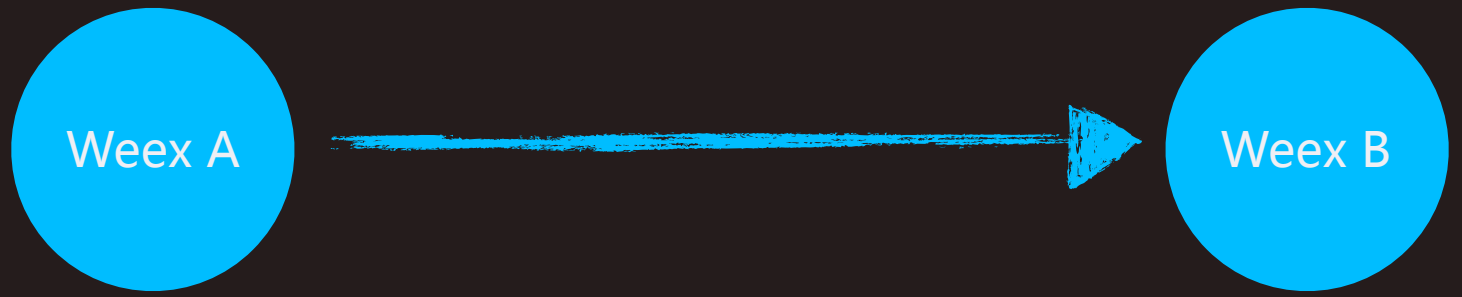
封装module的方式：gmmmodule.load(url, callback)

url采用常见的拼接方式&透传给Native，并解析去用

如何解决数据通信问题?

WeexA -> Weex B（weex实例页面间的跳转传递数据）

采用内建模块storage




```
storage.setItem(key, JSON.stringify(val), event => {
    if (event.result === 'success') {
        callback && callback()
    }
})
```

```
storage.getItem(key, event => {
    if (event.result === 'success') {
        callback && callback(JSON.parse(event.data))
    }
    storage.removeItem(key, (event) => {})
})
```

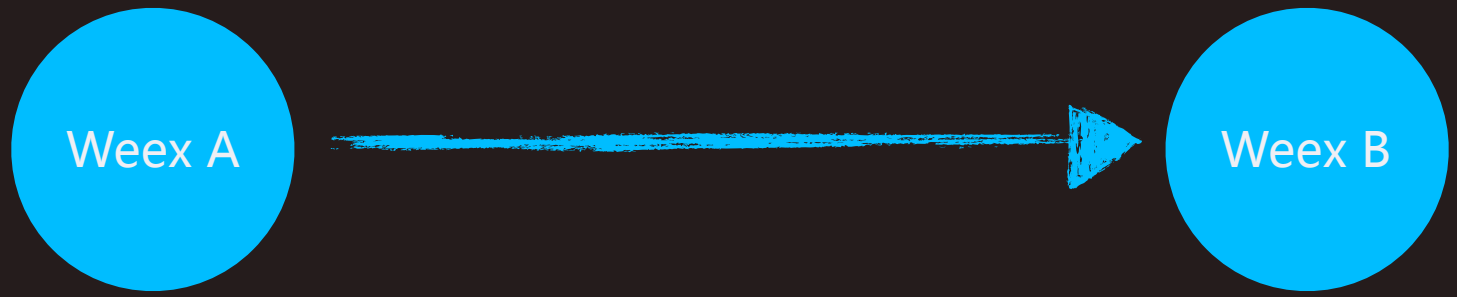
注意：接收到数据后清除一下

WEEX

如何解决数据通信问题?

WeexA -> Weex B（weex实例页面间的跳转传递数据）

viewappear与viewdisappear



viewappear事件将会在打开新页面时被触发。
viewdisappear 事件会在页面就要关闭时被触发

这两个事件关注的是整个页面的状态，所以它们必须绑定到页面的根元素上

WEEX

如何解决数据通信问题?

WeexA -> Weex B -> Weex A

B页数据会回传给A页

如何解决数据通信问题?

WeexA -> Weex B -> Weex A

来个形象点的场景看

点击选

选中一项
pop并回传

如何解决数据通信问题?

WeexA -> Weex B -> Weex A

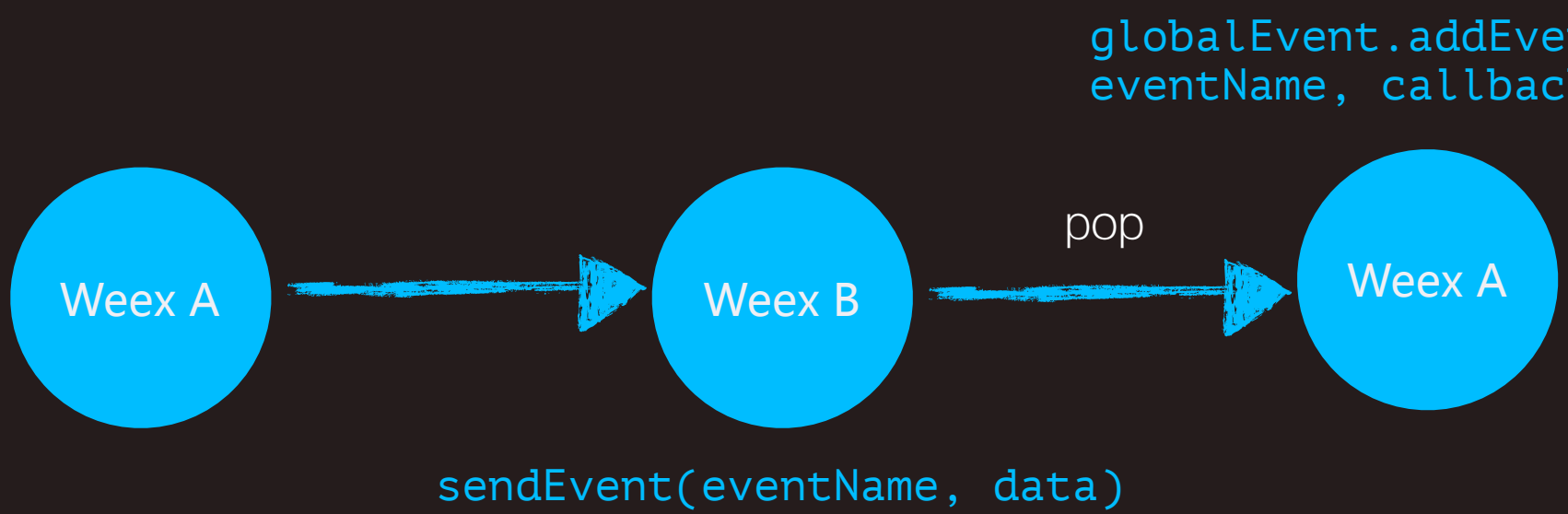
BroadcastChannel是weex实例间通信的解决方案

可惜仅.we版本可用，vue版本暂不支持!

如何解决数据通信问题?

WeexA -> Weex B -> Weex A

幸好Native提供fireGlobalEventCallback，我们在module里封装一个
sendEvent(eventName, data)
该方法用于触发实例的fireGlobalEventCallback方法



如何解决数据通信问题?

WeexA -> Native -> Weex B

- gmmmodule.load(url, callback)，callback做了回调，在回调中push到B
- 封装自定义module，钱包支付模块如下所示:

```
@JSMethod(uiThread = true)
public void pay(String data, final String cb) {
    Gson gson = new Gson();
    Map map = gson.fromJson(data, Map.class);
        new GBaoServiceApi((Activity) mWXSDKInstance.getContext()).pay(map, new GBaoServiceApi.GBaoApiCallbac
{
        @Override
        public void callback(String resultStatus, String msg) {
            Gson gson = new Gson();
            Map<String, String> map = new HashMap<>();
            map.put("return_code", resultStatus);
            map.put("return_message", msg);
            WXBridgeManager.getInstance().callback(mWXSDKInstance.getInstanceId(), cb, gson.toJson(map));
        }
    }, false);
}
```

样式如何做到复用?

一般而言，vue模块化会把css都内嵌在.vue中

但是，为了复用以促使UI的敏捷开发，所以采用如下方式（多亏了css-loader）：
我们正在做的是Weex UI组件…

```
<template src="./template.html"></template>
<style src="./style.css" scoped></style>
<style src="Styles/base/layout.css"></style>
<style src="Styles/base/util.css"></style>
<style src="Styles/base/icons.css"></style>
<style src="Styles/components/form.css"></style>
<style src="Styles/components/pics.css"></style>
<style src="Styles/components/dialog-content.css"></style>
<style src="Styles/components/title.css"></style>
<style src="Styles/components/picker.css"></style>
<style src="Styles/modules/list_5.css"></style>
<style src="Styles/modules/pic.css"></style>
```

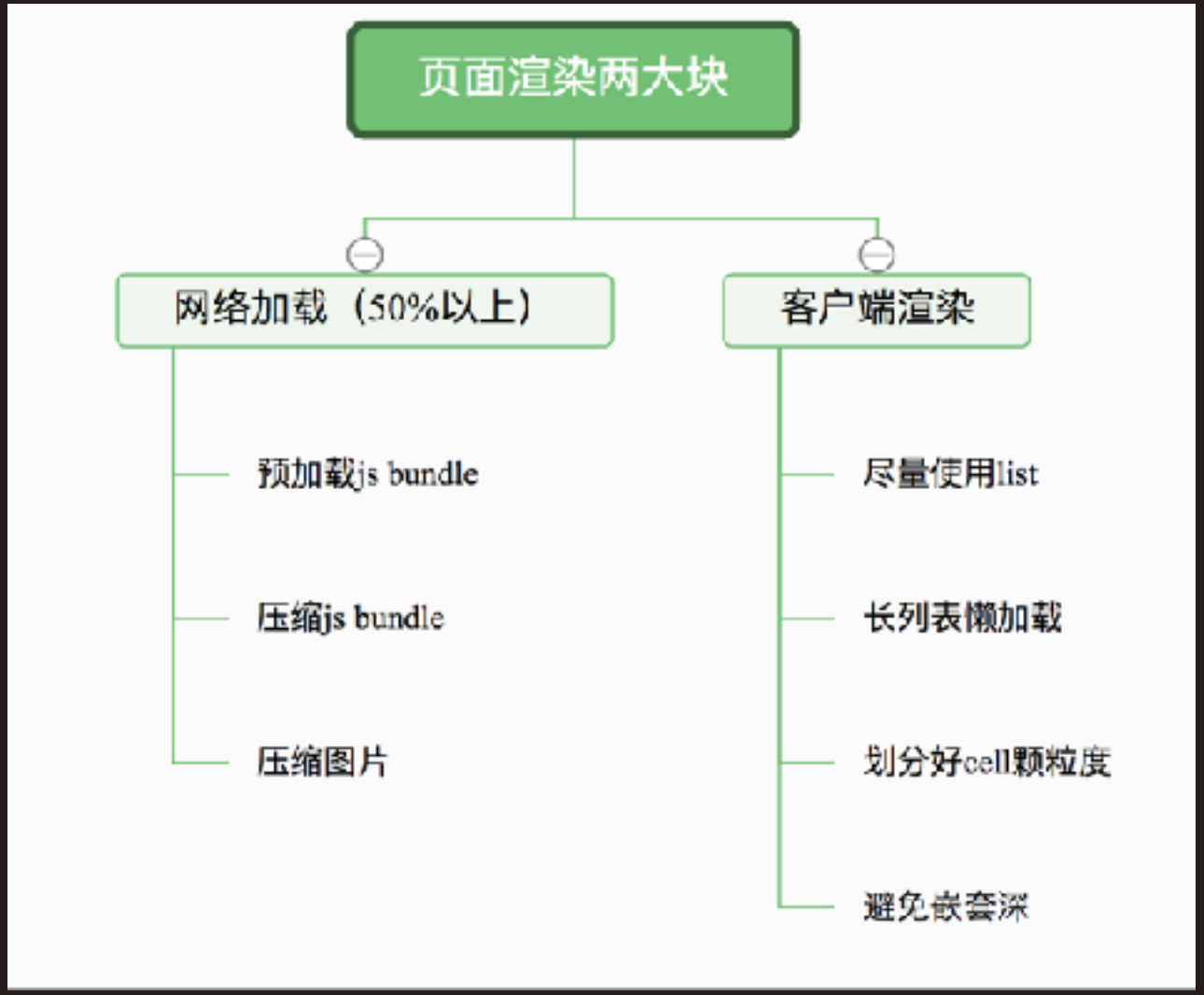
如何充分利用Native能力?

我们目前运用最多的是Native自定义module，复用一些封装良好的原生组件

take photo / postImage（图片上传）
share（分享）
goToBigImageList（图片列表大图全屏预览）
login（通用登录框组件）
pay（Gbao支付）
checkSms（短信验证组件）
selectDuration（时间要求控件）
…

性能优化

我们做的优化点有以下几点：

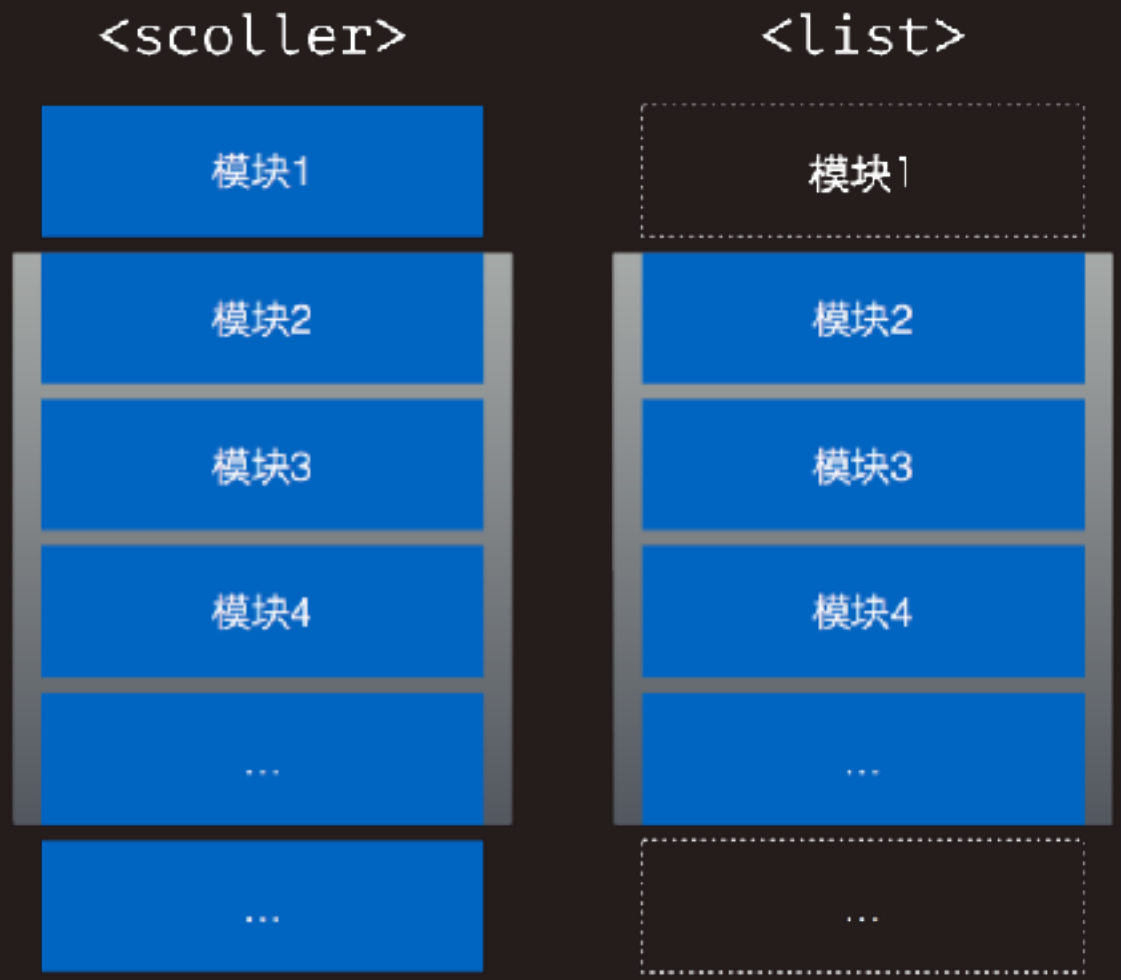


性能优化

列表渲染，首选<list>

<scroller>所有子组件
一次性渲染

<list>渲染可视区域子组件
子组件移出可视区域后
内存收回



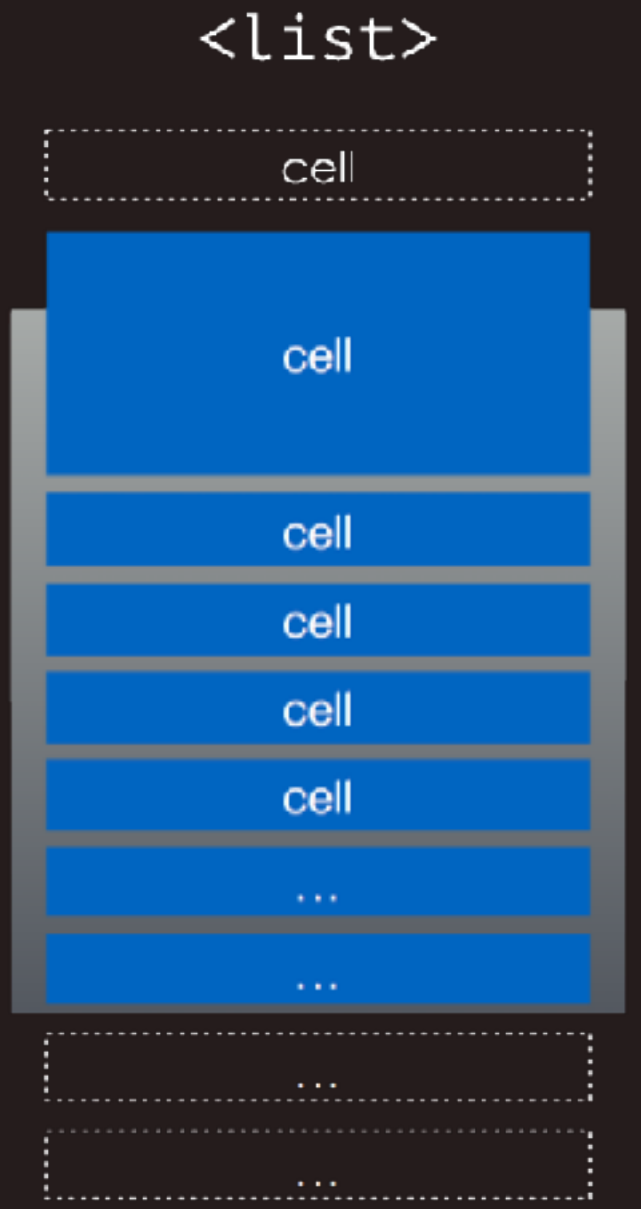
WEEX

性能优化

细粒度拆分<cell>

颗粒度越小、内存利用率越高

cell与cell之间独立渲染
且cell默认以tree模式解析
粒度越小，内容呈现越快



WEEX

踩坑填坑

Dialog点透问题

@click=“cancel”

点击这触发cancel
不科学啊！

e.stopPropagation()可解决问题

WEEX

踩坑填坑

图片无法确认大小的时候如何上下左右居中?

WEEX

踩坑填坑

图片无法确认大小的时候如何上下左右居中?

- 我们希望服务端返回的图片路径携带宽高信息，如:
  http://pics.sdoprofile.com/sdo4/M00/AA.jpg?size=184x184
- Native正好有现成的组件，而且可以滑动切换图片

踩坑填坑

样式上的一些不便之处

- 无法用样式绘制半透明的三角形（可采用图片替代）
- text组件上的大小、颜色样式不能继承（只能一个个都写上）
- 根据750px进行缩放，会有浮点级别的误差（用scale，deviceWidth进行计算）
- 不支持z-index层级关系（层级更高的元素往靠后）
- 定位元素超过容器边界，在 Android 下，超出部分将不可见（放在父辈容器中）
- 动态绑定class属性，vue的写法 :class={'btn54-text-orange': it.isStrong}
  （Weex的写法 :class="[it.isStrong ? 'btn54-text-orange' : '']"）
- 不能用背景图片（用<image>或者icon font来替代）

踩坑填坑

其他一些坑

- 长页面在Android配置较低的机子上渲染略慢（本地预先缓存、颗粒度控制、避免标签嵌套过深）
- 仅有Flexbox、position、盒模型布局，略显局限（充分利用Flexbox，扬长避短）
- 三端一致性还是不容易搞（仅做Android，iOS两端还是很开心滴）
- 组件不够丰富（造轮子，充分利用Native已有的能力）
- 本地开发体验有点捉急（十八般武艺都使出，Weex Playground做样式和交互，模拟器做业务逻辑）
- 遇到问题，可以参考的资料显少（多问多想多试）

QA

谢谢大家！

https://weex.apache.org