

Weex In GMM



就是他！

时间-2017年7月8日

地点-上海张江博云路111号浦软爱酷空间

主办方-iTechPlu

李永亮
*kissy*小鬼

 盛大游戏® 游麦中心-前端
微信号: *liang-love-rui*



概要

- 1、初期选型
- 2、小试牛刀
- 3、大规模应用
- 4、性能优化实践
- 5、踩坑填坑

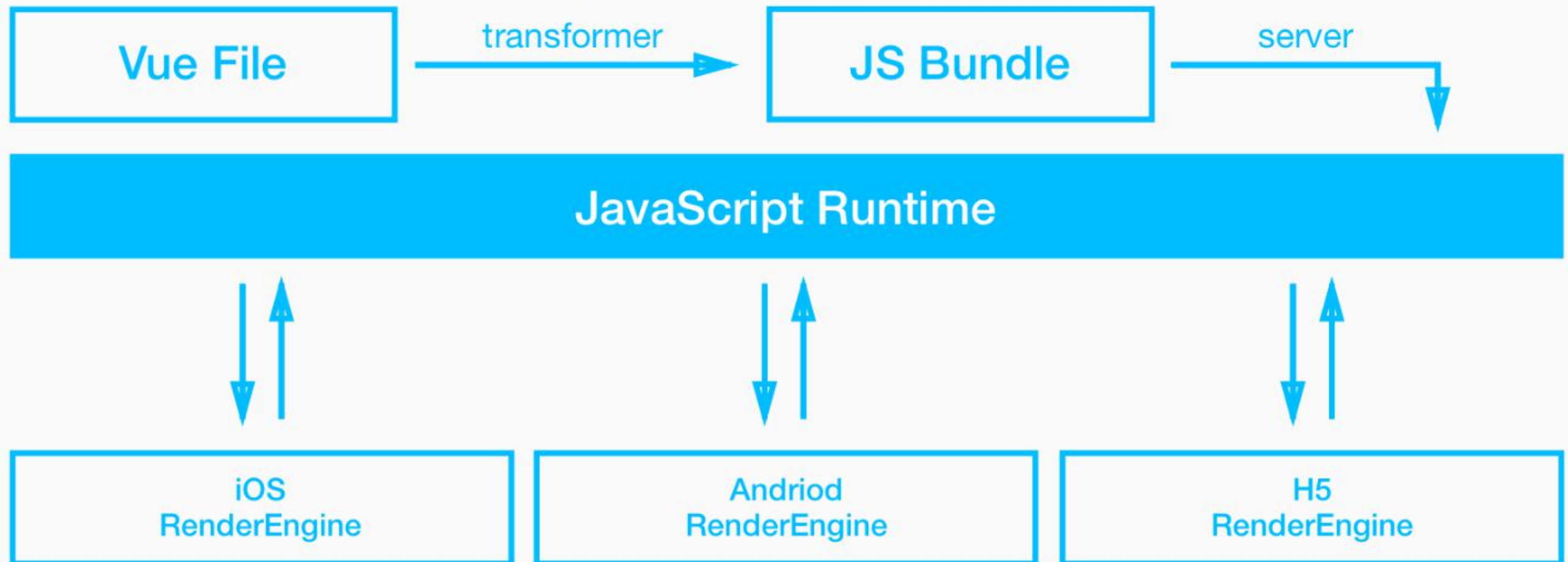
聊一聊技术选型



公司CTO与大前端分会倡导下，自上而下的期许



WEEX



选中Weex之理由

一、提升开发效率、交付速度

*Web*技术

一次编写，多端运行

*Vue*语法简单，*APP*童鞋学习成本低

易复用现有的*Vue*组件与*APP*组件功能

选中Weex之理由

二、高性能、优秀的用户体验

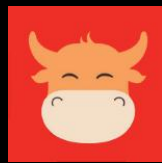
体积小，下载快

渲染速度快，铸造流畅的原生体验

对加载时间和资源占用深度优化

选中Weex之理由

三、大厂相对靠谱，*N*多APP在使用，可放心



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、千千氏、盛大游戏 ...

聊一聊脚手架



在项目中小试牛刀

废话不多说，先看东西！

系统消息页



商品管理

[查看详情](#)

您收藏的【[龙神188级]花瓶 永恒宠物大甩卖。泡泡伪天真，永恒宠物便宜甩】信息已经更新，请登录G买卖查看。

🕒 07-04 11:57

账号交易

[查看详情](#)

【[法师48级]送龙手对 紫对 铃铛】下单成功，请尽快付款，超时将自动取消交易，订单号为【70005902】。

🕒 06-29 17:56

账号交易

[查看详情](#)

【[战士53级]53级双战号】下单成功，请尽快付款，超时将自动取消交易，订单号为【80006852】。

🕒 06-29 17:52

信用评价

[查看详情](#)

您完整填写个人资料，成功获得买家信用10点和卖家信用10点。

🕒 06-28 20:44

小经验、小体会

一、*Weex SDK*版本升级，高版本优于低版本

Q: *sdk*版本0.7.0, *list*内<loading>看不到文字

Weex sdk 0.7.0升级到0.10.0后<loading>更佳

目前我们采用的是*weex_sdk:0.11.0*

小经验、小体会

二、登录态

Q: 服务端接口如何携带登录态?

客户端把*Ticket*传递给*weex*, 请求带上登录态, 客户端封装*cookie*给*weex*请求

```
@JSMethod(uiThread = true)
public void get(String url, final String cb){
    ServiceApi.sendHttpRequestForWeex(url, new MhhReqCallback<String>() {
        @Override
        protected void onSuccess(String result) {
            WXBridgeManager.getInstance().callback(mWXSDKInstance.getInstanceId(), cb, result);
        }
    });
}
```


小经验、小体会

三、Weex到Native的跳转交互

Q: weex如何优雅地跳转到原生页并传参?

封装成module方法: `gmmmodule.load(url, cb)`

url如: `sdggmm://product_comment?book_id=1`

`cb`参数在有些场景中可以使用（如评价页）

小经验、小体会

四、*dotwe*改造为*vue*版本

Q: *weex*升级到*vue2.0*需要做什么？

*weex-toolkit*在1.0.1之后才支持*vue*项目

借助*weex-vue-migration*工具实现语法转换

再手动调整一番

规模化应用Weex

在游戏代练业务中21个页面采用weex



首先需要考虑的

Q: 采用单页面还是多页?

Q: 页面跳转存在什么问题?

Q: 如何解决数据通信问题?

Q: 样式如何做到复用?

Q: 怎么充分利用*Native*?

在页面研发模式的道路上一度误入爱的迷途

SPA?

我们深爱着SPA

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

爱得深，伤得也深

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

团队协作差

全家桶(*vuex+vue-router...*)学习成本高

有一些未知的问题

多页君，爱可以重来！

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

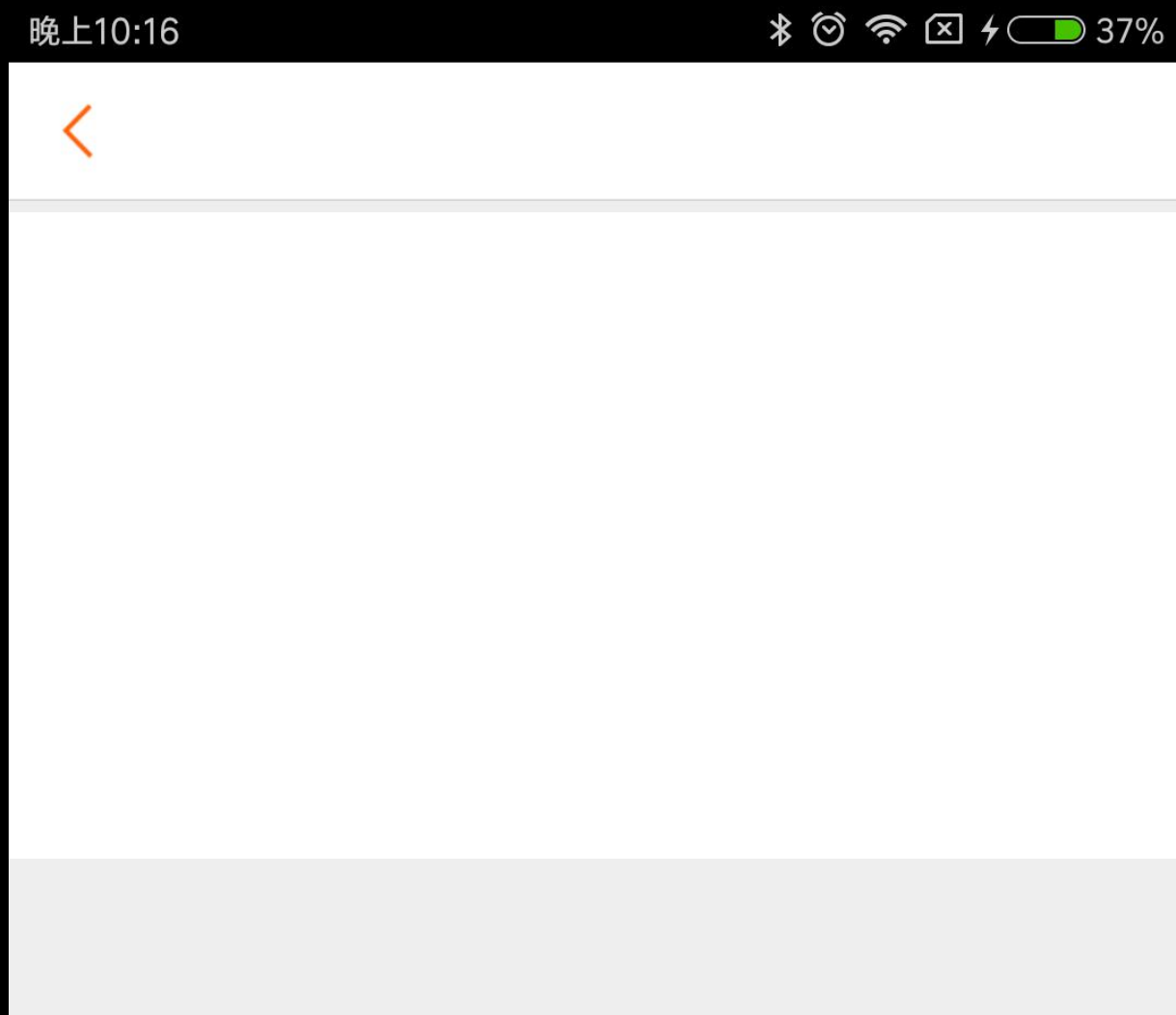
更好的内存管理，减少*Crash*的次数



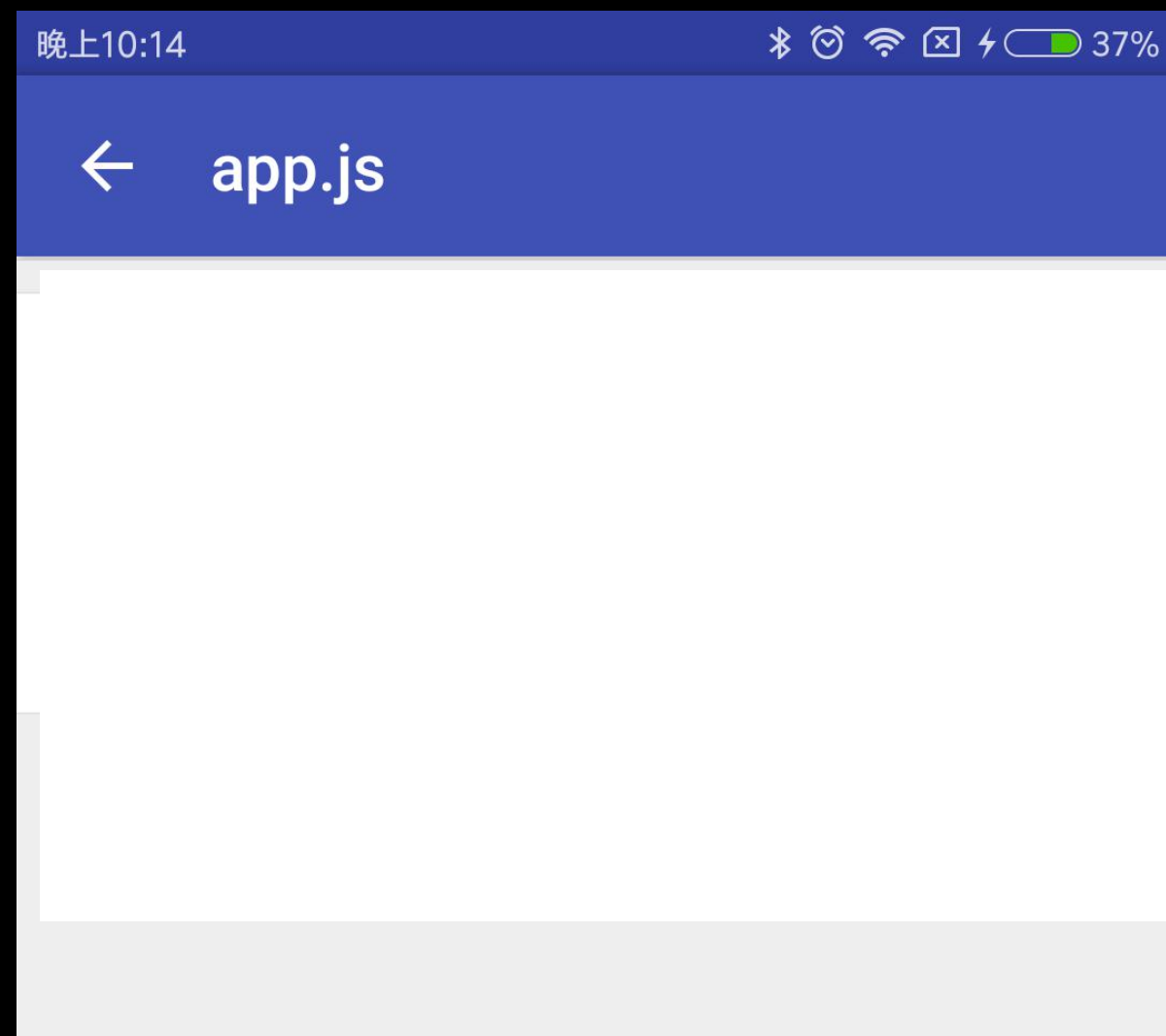
你不爱我了

页面间的跳转并非易事

我们期望这样子



结果却是这样子



页面间的跳转并非易事

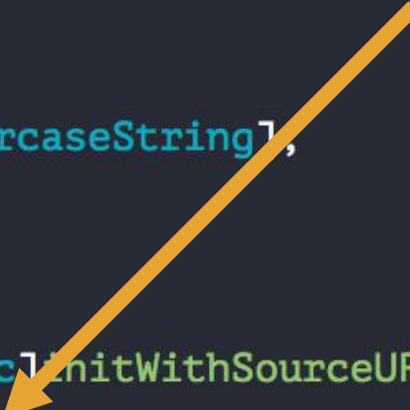
怎么办？重写SDK中navigator的push方法

```
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];
}
```



页面间的跳转并非易事

采用自定义`module`调用

```
gmmmodule.push({  
  animated: 'true',  
  url: utils.getBaseURL(vm) + `dailian/order/modifyDailian/app.js`  
})
```

全面数据通信方案

Native  Weex

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":"DLTFW100001000014993407798020450","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

全面数据通信方案

Native  Weex

若要推送消息给Weex页面，*globalEvent*可以帮上忙

Native

```
@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

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

全面数据通信方案

Weex  *Native*

gmmmodule.load(url,
callback)

*url*采用常见的拼接方式&透传给*Native*,
并解析去用

全面数据通信方案

Weex A  Weex B

storage module

weex A: storage.setItem(key, value, callback)

weex B: storage.getItem(key, callback)

全面数据通信方案

Week A  Week B

*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

全面数据通信方案



先说遇到的场景吧～

操作系统	安卓	>
渠道	微信(安卓)	>
区服	微信14区 庄子化蝶	>

点击选



选中一个
*pop*并填充

<

选择区服

● 区服

微信7区 洛神降临

微信8区 王者审判

微信9区 王者惩戒

微信10区 王者守御

微信11区 至尊王权

微信12区 火力压制

微信13区 无敌鲨炮

微信14区 自然意志

微信15区 庄子化蝶

微信16区 蝴蝶效应

微信17区 天人合一

微信18区 磁力屏障

微信19区 霸王护盾

微信20区 机关魔爪

完成

全面数据通信方案

*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！坑爹！



办法倒是有一个！

全面数据通信方案

*Native*提供*fireGlobalEventCallback*

```
@JSMethod(uiThread = true)
public void sendEvent(String eventName, String data) {
    List<WXSDKInstance> instances = WXSDKManager.getInstance().getWXRenderManager().getAllInstances();
    Gson gson = new Gson();
    Map map = gson.fromJson(data, Map.class);
    for(WXSDKInstance instance : instances) {
        instance.fireGlobalEventCallback(eventName, map);
    }
}
```


全面数据通信方案

刚刚场景中，区服选择后，把选中传给商品上架页

区服页完成

```
// 给父组件更新区服数据用  
utils.sendEvent('area_group_data', selected_area_group)
```

区服页回传

```
const globalEvent = weex.requireModule('globalEvent')  
globalEvent.addEventListener('area_group_data', (json) => {  
  vm.game_app_os = json.game_app_os  
  vm.game_operator_id = json.game_operator_id  
  vm.area_id = json.area_id  
  vm.group_id = json.group_id  
  vm.area_name = json.area_name  
  vm.group_name = json.group_name  
})
```

样式的复用

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，所以采用第二种方式如下（多亏了*css-loader*）：

```
> app.vue ×  
  
<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>
```

巧妙滴利用*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）

...

Weex中的vue 2.0

双向绑定

组件间通信

数据驱动

组件颗粒度

简说区服组件



双向绑定的两种方式

*v-model*方式，看栗子

父

```
<gmm-loading v-model="showLoading"></gmm-loading>
```

子

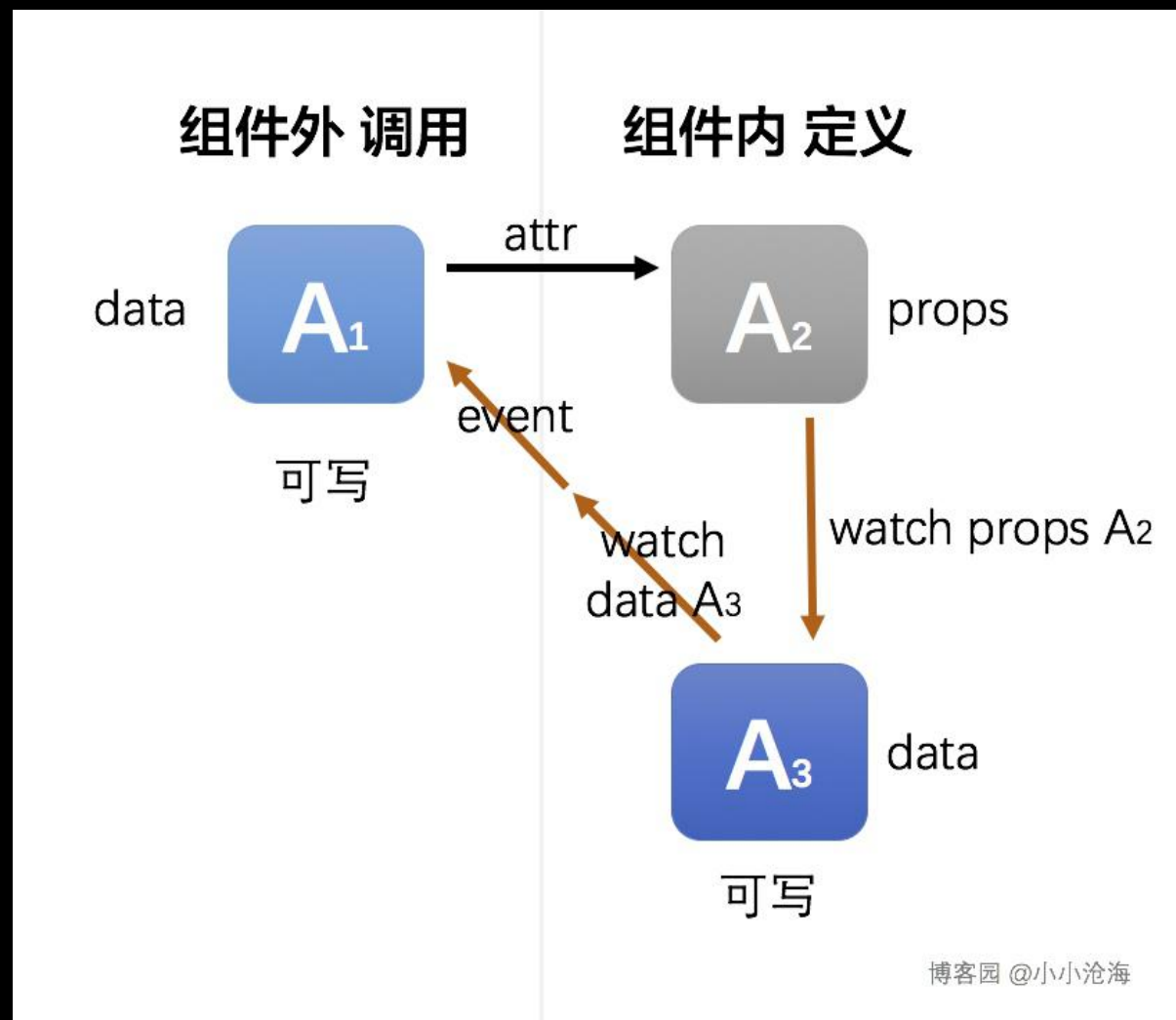
```
<div class="loading" v-if="value">  
  <text>正在加载中...</text>  
</div>
```

加载完触发: *this.\$emit('input', false)*



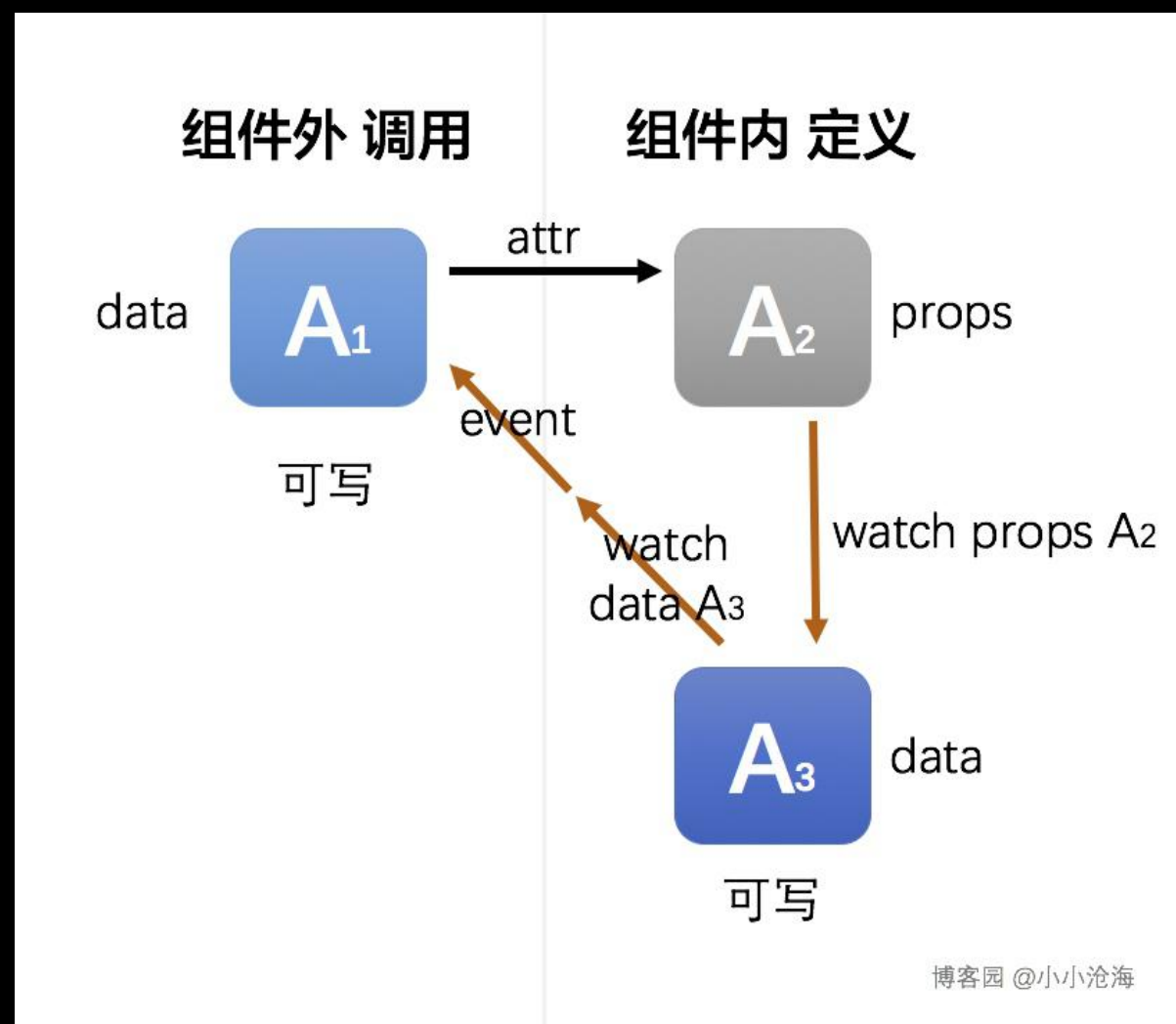
双向绑定的两种方式

三部可完成, *dialog*采用这种方式



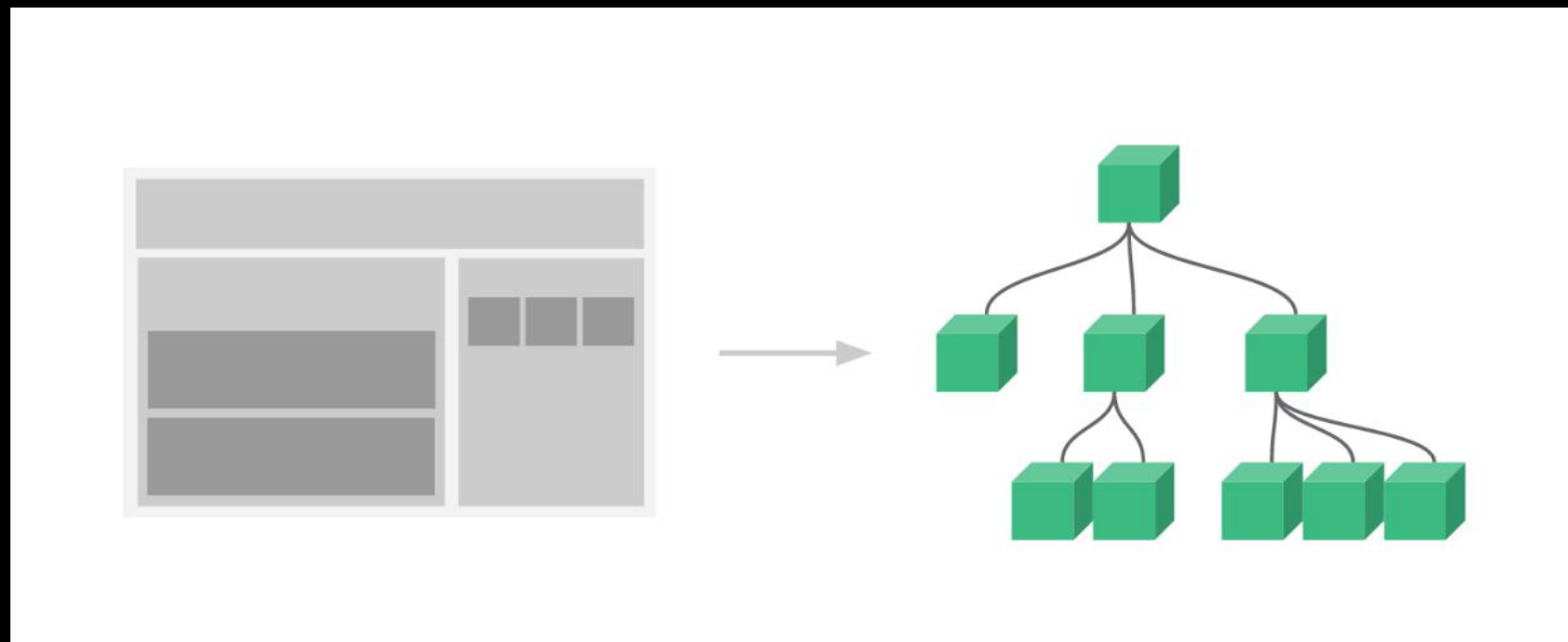
组件间通信

两个关键词 *props*、*\$emit* 事件传递



组件颗粒度划分

并非越细越好、也并非要大而全，讲究个度



简单说下区服组件

这是一个相对复杂的业务组件（代码）

操作系统	安卓	>
渠道	微信(安卓)	>
区服		>

+

晚上10:36 38%

< 选择区服

● 区服

- 微信7区 洛神降临
- 微信8区 王者审判
- 微信9区 王者惩戒
- 微信10区 王者守御 ☒
- 微信11区 至尊王权
- 微信12区 火力压制
- 微信13区 无敌鲨炮
- 微信14区 自然意志
- 微信15区 庄子化蝶
- 微信16区 蝴蝶效应
- 微信17区 天人合一
- 微信18区 磁力屏障
- 微信19区 霸王护盾
- 微信20区 机关魔爪

完成



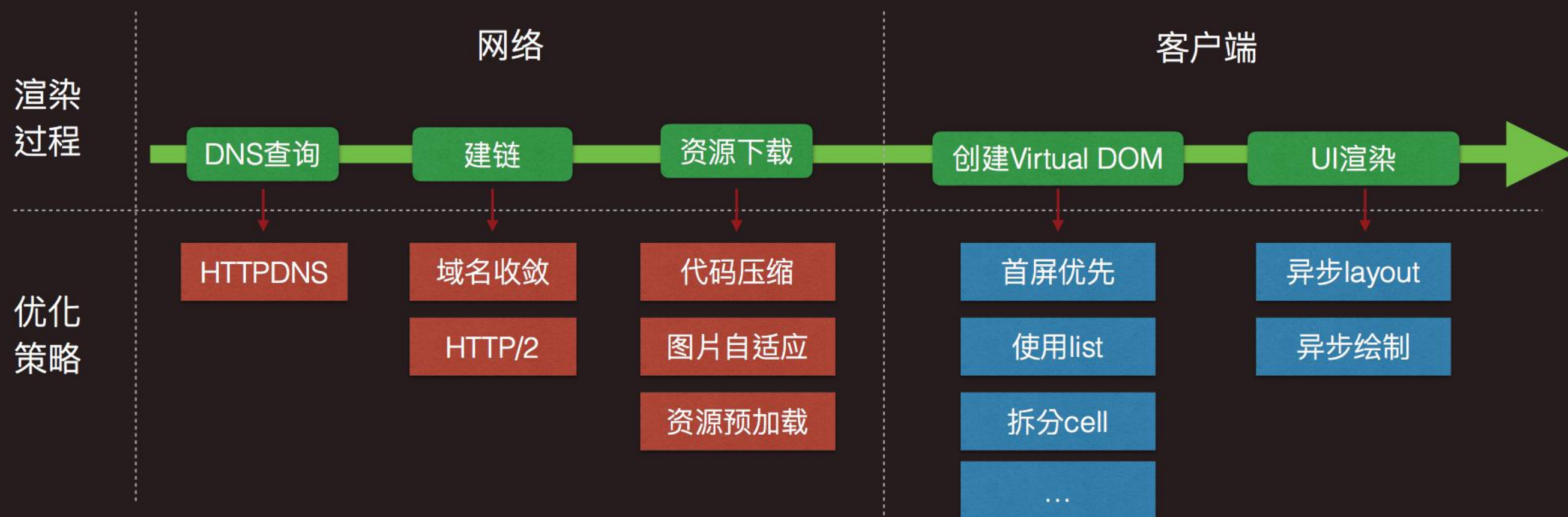
性能优化实践三五点

我们希望秒开：在一秒内首屏达到可交互状态



页面加载时间 + 首屏渲染时间 $\leq 1s$

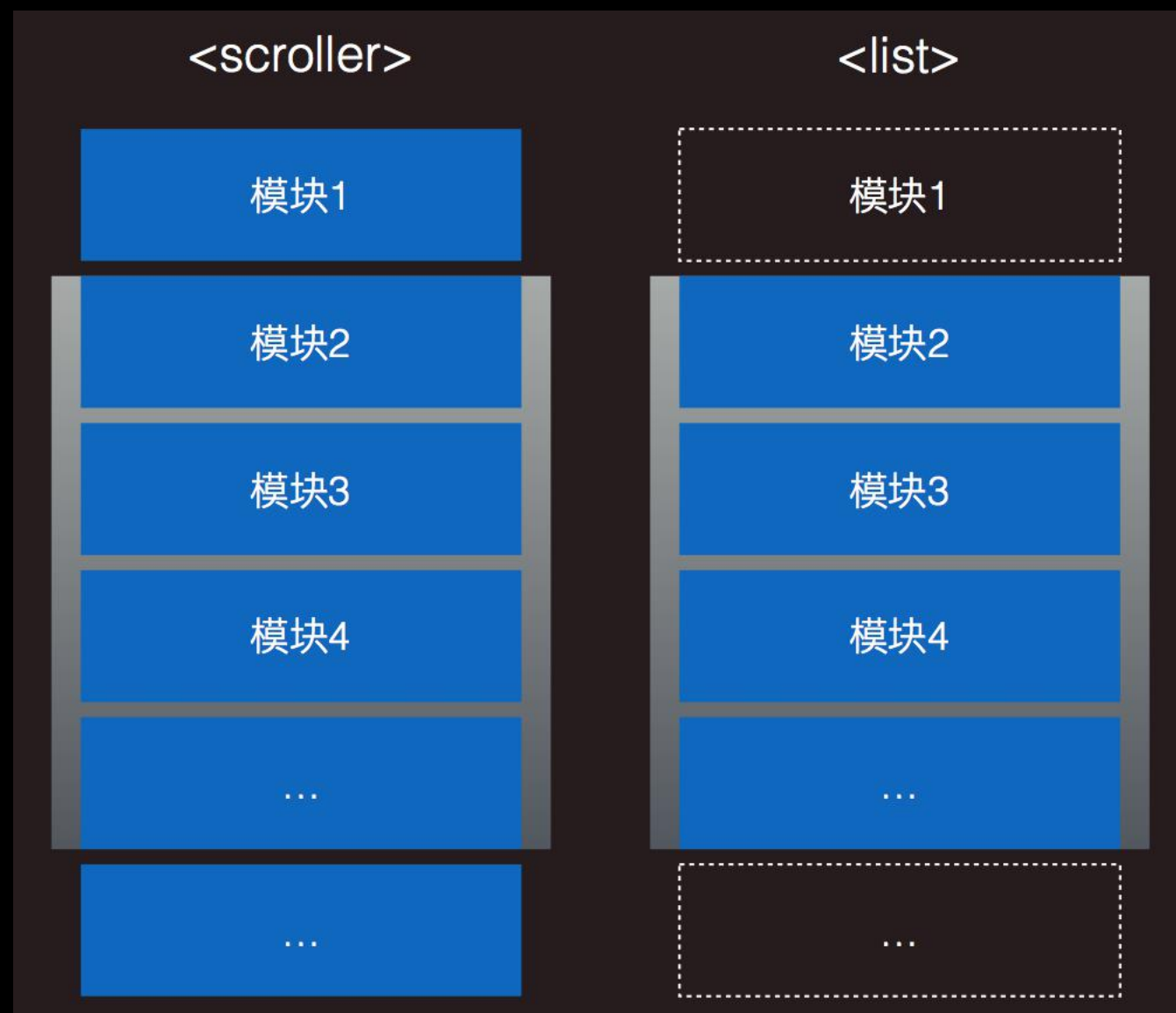
优化策略：减少网络下载时间，加速页面渲染



一、列表渲染，首选<list>(订单详情demo)

<scroller>所有子组件
一次性渲染

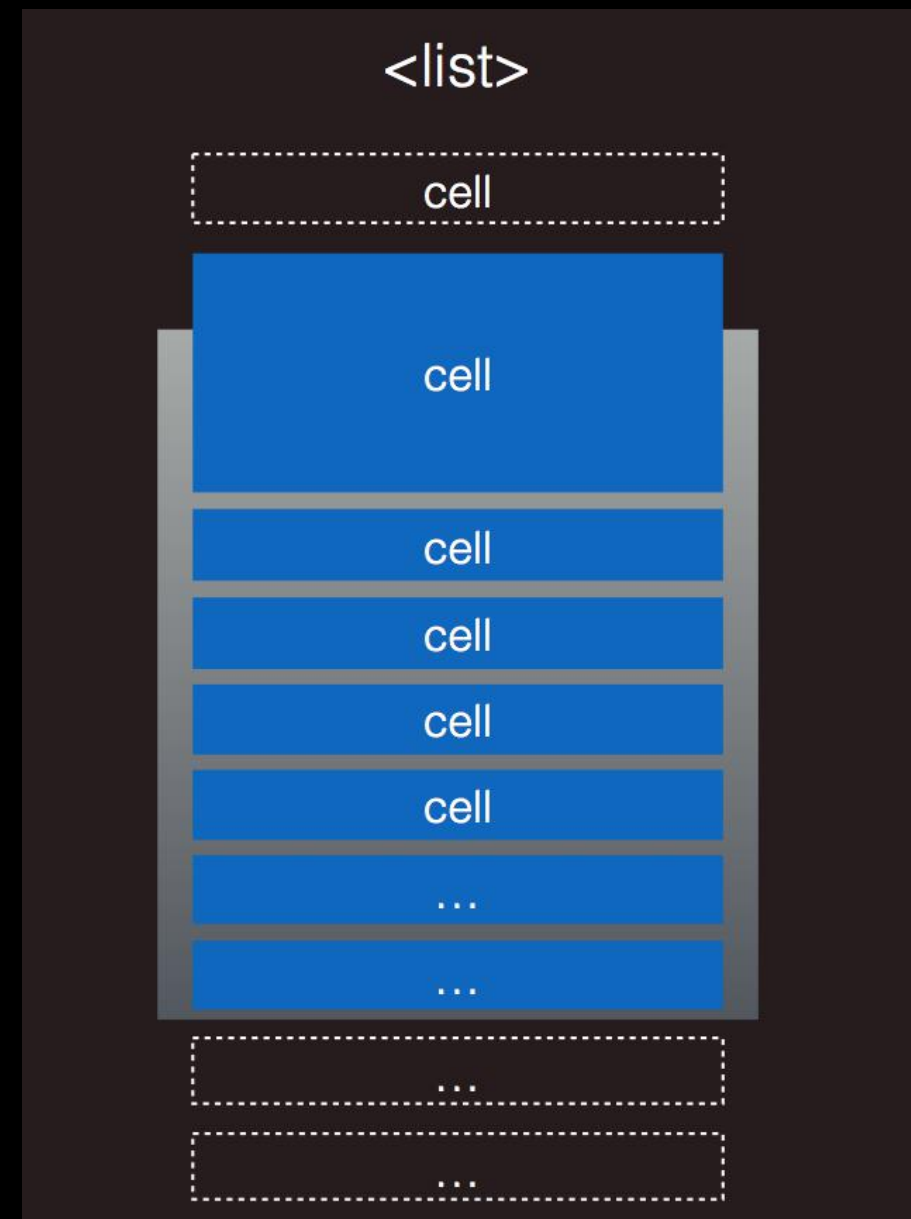
<list>渲染可视区域子组件
子组件移出可视区域后
内存收回



二、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

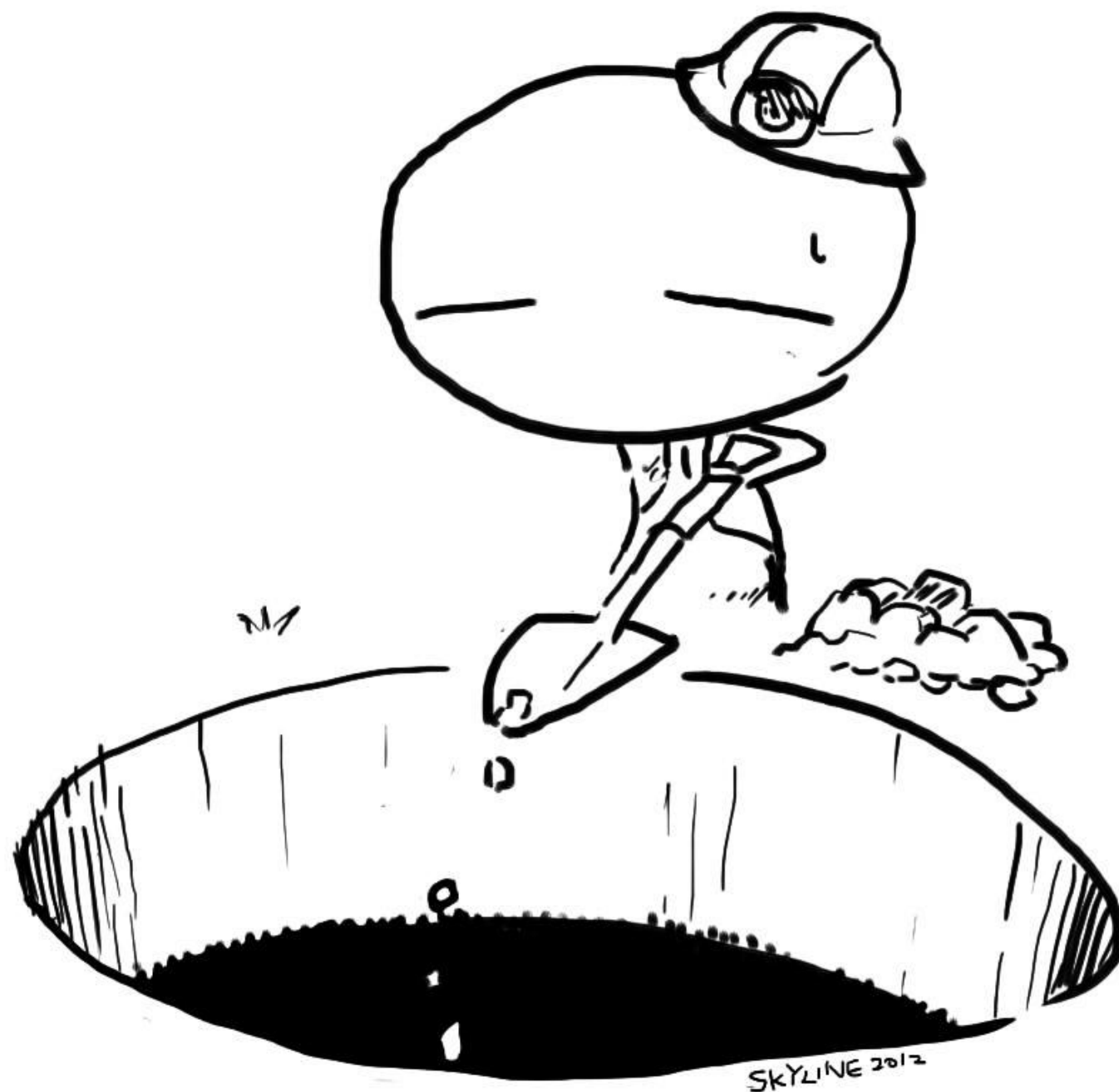


三、避免过多*dom*层级嵌套

四、首屏渲染 + 加载更多（<*loading*>）

五、资源预加载、客户端下载处理(*zip*下载)

有坑填坑



一号坑: *vue*数组内部对象更新, 未及时渲染

时间要求		>
获得物品		>
取消	选择时间	确定
0天	1小时	
1天	2小时	
2天	3小时	

场景一

```
selectDuration (groupId, fieldId) {
  let vm = this
  utils.selectDuration(vm.dayHourStr, (ret) => {
    let secondsStr = ''
    vm.show_hide_template.forEach((group) => {
      if (group.id === groupId) {
        group['fields'].forEach((field) => {
          if (field.id === fieldId) {
            secondsStr = utils.toSeconds(ret.day, ret.hour)
            vm.dayHourStr = ret.day + ret.hour
            field.value = secondsStr !== 0 ? secondsStr : ''
          }
        })
      }
    })
  })
},
```

方法一: *\$set*方法

```
selectDuration (groupId, fieldId) {  
  let vm = this  
  utils.selectDuration(vm.dayHourStr, (ret) => {  
    let newShowTemplate = {}  
    let secondsStr = ''  
    vm.show_hide_template.forEach((group) => {  
      if (group.id === groupId) {  
        newShowTemplate = group  
        newShowTemplate['fields'].forEach((field) => {  
          if (field.id === fieldId) {  
            secondsStr = utils.toSeconds(ret.day, ret.hour)  
            vm.dayHourStr = ret.day + ret.hour  
            field.value = secondsStr !== 0 ? secondsStr : ''  
          }  
        })  
      })  
    })  
    vm.show_hide_template.$set(0, newShowTemplate)  
  })  
},
```

方法二：数组*push*方法（有点黑科技）

```
selectDuration (groupId, fieldId) {  
  let vm = this  
  utils.selectDuration(vm.dayHourStr, (ret) => {  
    let secondsStr = ''  
    vm.show_hide_template.forEach((group) => {  
      if (group.id === groupId) {  
        group['fields'].forEach((field) => {  
          if (field.id === fieldId) {  
            secondsStr = utils.toSeconds(ret.day, ret.hour)  
            vm.dayHourStr = ret.day + ret.hour  
            field.value = secondsStr !== 0 ? secondsStr : ''  
          }  
        })  
      }  
    })  
  })  
  vm.show_hide_template.push()  
},
```

—————> 一位同事告诉的

二号坑: *Dialog*点透问题

场景二

基本信息

标题 我一厢情愿

我的单价 ¥ 1.00

库存 25件

手续

预估

保证

代练

代练

加密信息

确认发布代练

标题: [代理服务]我一厢情愿

代练单价: ¥1.00

库存: 25件

预冻保证金: ¥5.00

预估手续费: ¥1.00 (以实际成交价为准)

取消 确认

`@click="cancel"`

点击这触发`cancel`
不科学啊!

`e.stopPropagation()`

三号坑: *input*输入时, 无法实时删除么?

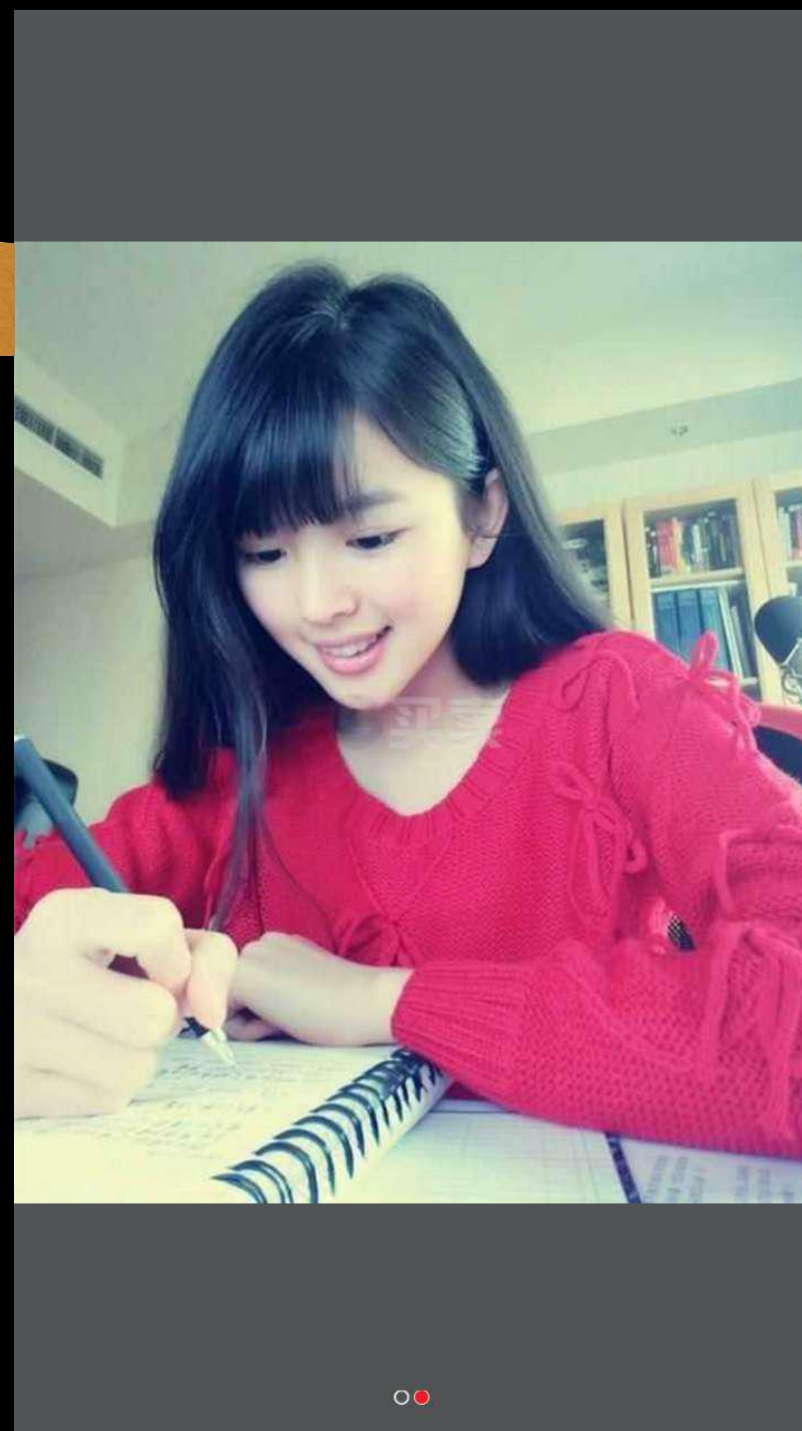
基本信息		
标题	填写代练服务的标题	
代练单价	¥ 1.00	
库存	20	件
手续费标准		
预冻保证金	¥ 4.00	
保证金标准		

```
onMoneyInput (event, fieldName) {  
  let vm = this  
  let text = ''  
  let reg = event.value.match(/\\d+\\.?\\d{0,2}/)  
  if (reg != null) {  
    text = reg[0]  
  }  
  vm[fieldName] = text  
},
```



```
onMoneyInput (event, fieldName) {  
  let vm = this  
  let text = ''  
  let reg = event.value.match(/\\d+\\.?\\d{0,2}/)  
  if (reg != null) {  
    text = reg[0]  
  }  
  vm[fieldName] = event.value  
  vm.$nextTick(() => {  
    vm[fieldName] = text  
  })  
},
```

四号坑： 图片无法确认大小的时候如何上下左右居中？



四号坑： 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/MOO/AA.jpg?size=184x184>

*Native*正好有现成的组件， 而且可以滑动切换图片

遇到的问题也许并非*weex*的问题
但*weex*依然存在诸多不完美

仅有 Flexbox 布局, text 无法嵌套, 难以实现长文当中样式的混合

No Cookies. 只能用 storage 来完成对应信息的存储

三端一致性不完全, 特别是 HTML5 支持不完善

CSS 和 Web 当中存在差异

DevTools 的成熟度不够. 热替换不够强大

相对 Web 而言组件丰富度不够

谢谢!

iTechPlus技术

“新前端实战营 开课啦”

Vue+Weex周末实战班
Vue+Weex

盛大游戏前端技术专家
李永亮 亲自授课

12999元

1期班：2017/07/22~23
2期班：2017/08/05~06
3期班：2017/08/19~20



上海长宁区中山西路179号
虹桥世家花园商务楼3楼

vue+weex周末实战班

李永亮

上海 浦东新区



扫一扫上面的二维码图案，加我微信

个人微信-咨询/交流