（一直想写，今天终于有机会了.....）

17年大概做了4个 Vue 项目，2个后台管理系统，一个纯移动端，一个内嵌在原生 App 的项目。中途还参与了一个 React 项目，自己也用 React 写过两个 demo（能更好的发现二者的异同点）。从而决定要写这篇文章，有写的不对的地方，求大佬亲喷。（默认大家都掌握了Vue, Vuex, Vue-router及其相关周边库）

一、从拿到原型图到UI图的一些准备工作

项目是要内嵌于原生 App 中，所以分析需求的时候，先得确定哪些东西是要交给原生来负责。

（PS：项目是一款知识付费的类型的 Hyprid App）

首先 H5 和原生 WebView 通信的原理，是在window下挂载一个对象，这里直接贴代码吧

```
const postMessage = (method, message) => {
 if (window.AndroidWebView) {
   // Android
   if (message) {
     window.AndroidWebView[method](message)
   } else {
     window.AndroidWebView[method]()
   }
 } else {
   // iOS
   window.webkit.messageHandlers.a.postMessage({
     body: {
       method,
       message
     }
   })
 }
}
```

接着将这个方法挂载到 Vue 下，就可以全局使用啦(PS：记住 message 里面的类型一定要是String)

```
Vue.prototype.$postMessage = postMessage
// 使用的时候很简单
const methodName = xxx
const toAppJson = {}
this.$postMessage(methodName, JSON.stringify(toAppJson))
```

比如调起原生的支付、分享，跳转原生页面，还有一些特殊的功能，也需要两边相互配合才可以实现，比如自动播放，自动聚焦的同时唤起键盘。不仅 web 端可以去调原生的方法，反过来也是可以的。

在支付成功，取消，失败后，或者分享后，来调用web端的方法

```
window.payCallback = (type) => {
  switch (type) {
    //...跳转路由
  }
}

window.onShare = (userId, type) => {
   // 给分享成功的用户加积分
}
```

还有最重要的一个功能，大多数 App，都有一个注册登陆的功能，但是内嵌的我们只需要一个‘假授权’登陆，即原生端将用户信息，存在 cookie，localStorage 里，取出来以后做加密后，请求登陆验证即可。这里不建议把用户信息拼接在URL后面，因为在安卓手机下，用户断网访问 Webview 会暴露整个 URL 信息出来。而且两者有个细微的区别，就是在取参数的时候，前者，必须等待整个 Webview 加载完成之后，才可以取到，所以我需要加一个延迟。

开发内嵌在 web，可能会遇到意想不到的 bug，所以你需要你能在手机上查看的控制台。（公司大佬推荐给我的，超级好用）

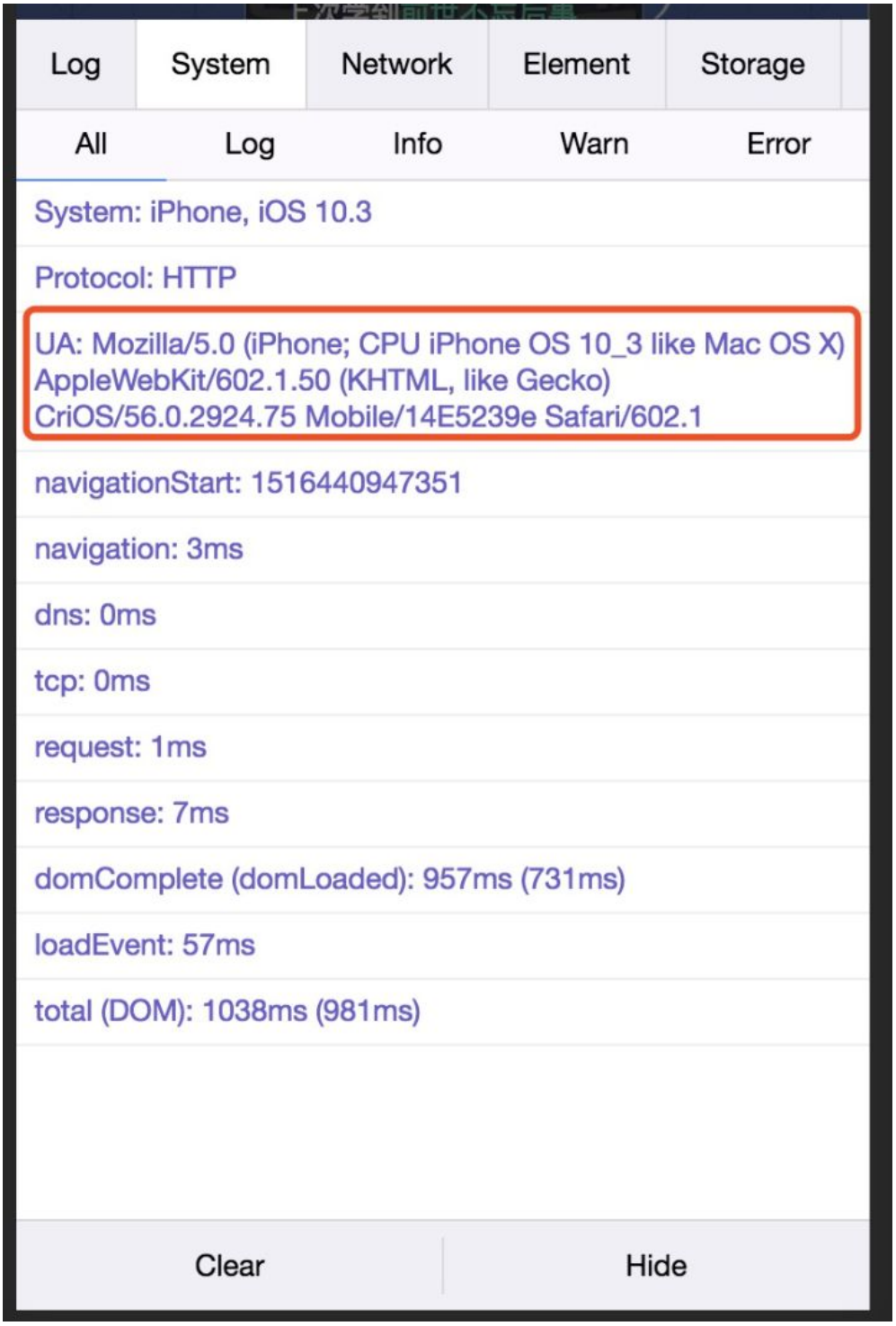
```
// 加载控制台
export const loadScript = (url, callback) => {
 const script = document.createElement('script')
 script.onload = () => callback()
 script.src = url
 document.body.appendChild(script)
}

loadScript(
'https://res.wx.qq.com/mmbizwap/zh_CN/htmledition/js/vconsole/3.0.0/vconsole.min.js',

 () => {
// eslint-disable-next-line
new VConsole()
 })
```

复制以上代码，在 main.js 引入，在手机上运行，即可出现如下按钮。

点开它，你就可以看的这个画面



红色框框标注，有什么用，我先卖个关子，后面详细说明...

二、技术栈，插件库，CSS方案选择

知识付费主要的功能是音频播放，数据搜集，统计，展示。付费用户和非付费用户所展示的界面，播放器的几种状态，用户的留言，评论，点赞等等...

首先说一下UI库的选择，移动端截止目前可供选择的还是蛮多的：

‘YDUI ‘（一只基于Vue2.x的移动端&微信UI。 -YDUI Touch）

最新版本，还处于公测阶段。我也是最近一段时间才听说这个款，并没有深入使用过，只是简单看过文档，有做个项目的小伙伴可在评论区说一下，另外按需加载配置起来略复杂。

‘有赞UI ‘（Vant - 有赞移动端 Vue 组件库）

第一次了解这个库，是在掘金上（如果没记错的话），有PC版（基于React）、移动端以及小程序版。最近有两个小项目，正在使用它开发。周四提了个issue，作者也是积极修复，立马就迭代了一个版本，这波必须打call。

采用目前主流的`babel-plugin-import`插件实现按需加载，不需要额外引入css，样式很好看。组件定位也很明确，就是帮助你迅速搭建一个商城，强烈推荐。

‘mint-ui ‘(mint-ui documentation)

饿了么团队出品，按需加载使用得是`babel-plugin-component` ，在前端发展如此迅速的今天，可以说是比较‘古老’的组件库，样式也不够炫。据知情人士不可靠消息称，饿了么团队今年会重新开发一套移动端的组件。

‘妹纸UI ‘（Muse-UI）

摸着良心说，这绝对是我目前使用过Vue UI库里，颜值最高的，没有之一。整体设计偏像国外的风格。而且感觉它不只针对移动使用，还能用于PC端。文档也写的非常漂亮。非要说一个缺点嘛，就是按需加载配置起来很麻烦。我折腾了一天，并没有成功，移动端流量第一，没解决这个问题，所以我只好放弃，都怪我太菜，不会配置webpack。

‘WeUI ‘(VUX - 移动端Vue组件库)

所以，最后我选择了Vux作为这个款web App唯一指定合作伙伴，你要问我它好在哪里，我可能说不出来，但是，真的是一款为开发移动端项目的精心准备的礼物。

1. 细致入微的文档
2. 移除移动端点击延迟、添加 viewport meta、微信jssdk、异步加载组件、webvView常见问题、加密工具、防抖和节流、1-px、日期格式化等等这些。对于第一次开发移动端的新手来说，是非常全面的
3. 有很多成功的例子，有兴趣的同学可以去官网看看
4. 超级丰富的组件库，我没有具体统计过。但是如上列举的5个常用库，它的组件最多。

（PS：使用 Cell 组件里左侧 icon，在低版本的安卓手机会有 bug，具体要看那款手机用的Chrome内核是多少，反正版本 30 以下会出现bug，即子元素没办法撑开父元素）

Align-items

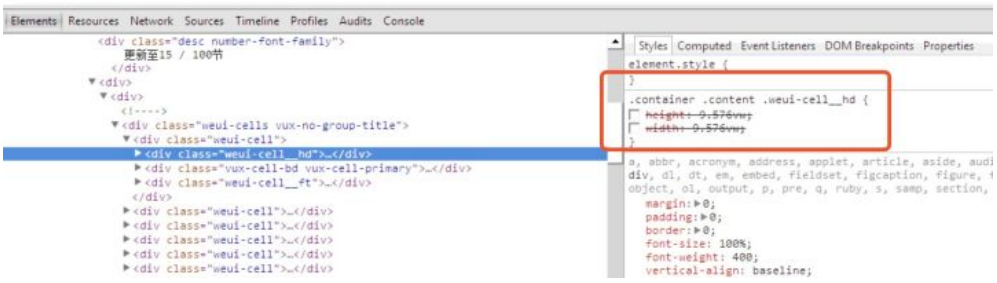
default	long long long longlong longlong longlong longlong longlong longlong longlong longlong long
flex-start	long long long longlong longlong longlong longlong longlong longlong longlong longlong long

Align-items

default	long long long longlong longlong longlong longlong longlong longlong longlong longlong long
flex-start	long long long longlong longlong longlong longlong longlong longlong longlong longlong long

第一张是高版本浏览器内核打开的效果，后面两张就是 Chrome30- 的效果了。这里说一下我是如何定位这个问题，因为这个 bug 只出现一台安卓系统4.4的手机，起初我以为是它的问题，但是呢有的又不会。这个时候，我上面提到那个真机调试控制台就展示效果了，我发现出现这个布局错乱的原因是由于浏览器内核造成的。但是只定位了一个大概，那么具体原因是啥，我们还得进一步分析。是 position 定位？flex 不支持？vw 单位不支持？都不是！

为了解决这个问题，终于让我找的了谷歌浏览 29，和 30 的版本的下载包。当时还不支持切换手机模式，终于让我发现原因，是没给父元素设高宽。可能是我 CSS 比较菜，或者是vux作者也没有想到这个点。



这次解决问题的经历还是挺有趣的，下次有机会面试别人，可以给他两部安卓手机和一部ios手机，让他调这个bug。看看他会如何分析，排查这个bug。
最后项目开发时间为一个半月，然后上了40天班。对于敏捷开发来说，在选取库的时候，尽量选择团队成员都了解，或者自己特别熟悉的。避免遇到突发情况，从而耽误开发进度。图表工具自然选择业界良心的 echarts(ECharts)，数据通信 axios （axios/axios），状态管理（Vuex 是什么？ • Vuex）后面我会详细说一下使用心得。

最后是 CSS 方案，一直以来都是用的 less，要为什么，我觉得它简单实用易上手。整个项目布局，都是以 flex 为主，以后有机会会尝试grid。还有一个重要的点，是如何做自适应布局，大家可能都习惯用 rem.js，或者是淘宝那套高清方案。但是结合具体项目，我们的产品面向的是安卓4.4+，iOS8.0+，考虑到是一款知识付费，以及开发成本，低于这个版本的用户，在体验上的不友好不在考虑范围之内，所以我选择了采用 vw 方案实现自适应。

```less
// color
@single: #4DDBB4;          // 提示文字单颜色
@gradient1: #52E99C;       // 渐变色1
@gradient2: #49CEC9;       // 渐变色2
@delete: #F13328;          // 提示文字和删除按钮

@generalText: #9494AE;     // 未选中状态、说明性文字、空白页icon
@generalBg: #EBF6FC;       // 空白页icon填充色

@weakBg: #FFFFFF;          // 内容区域底色
@weakLine: #E5E5E5;        // 分割线
@splitface: #f5f5f5;       // 分割面
@tagBg: #F7F9FB;           // 标签底色
@progressBar: #ECECEC;     // 白底上进度条灰色

@fontColor900: #333;       // 正文文字、联系人、标题
@fontColor600: #666;       // 次要文字
@fontColor300: #999;       // 辅助性文字
@fontColor100: #d8d8d8;    // 浅显说明文字、悬浮阴影

// font
@vw: 0.133vw;              // 基础单位约等于设计稿的1px
@space: 20 * @vw;          // 间距

@xxsFont: 3.2vw;           // 最小呈现字体        24px
@xsFont: 3.467vw;          // 辅助性文字          26px
@sFont: 3.733vw;           // 次要文字           28px
@font: 4vw;                // 正文内容           30px
@lFont: 4.267vw;           // 功能按钮、分类标题   32px
@xlFont: 4.8vw;            // 少数重要标题、按钮   36px
@xxlFont: 6vw;             // 文章标题 45px
```

设计稿是以为 iPhone6 为准的。那个 0.133vw 是在 100vw / 750 得到的。后面开发的时候，只需要引入这个 less 文件，然后把设计稿上标注由 px 替换成 * @vw 即可。

三、项目结构

以上两个点，我自己感觉讲的乱七八糟的，后面尽量保证清晰一点。（简单的就略过了）

src 下的目录结

```
|
|-- assets
|    |-- images          /* 图片 */
|    |-- less            /* 公共样式 */
|    |    |-- common.less
|    |    |-- variable.less
|    |    |-- reset.less
|    |    |-- index.less
|    |
|    |-- components      /* 组件 */
|    |-- filter          /* 格式化时间，数字，字符的函数 */
|    |-- mixins          /* 不解释 */
|    |-- router          /* 不解释 */
|    |-- store
|    |    |-- modules    /* 按模块拆分状态 */
|    |    |-- index.js
|    |-- svg             /* svg */
|    |-- views           /* 页面路由对应的组件 */
|    |-- App.vue
|    |-- dev-config.js   /* 开发阶段引入的配置文件 */
|    |-- main.js
|    |-- util.js         /* 工具函数封装 */
|    |
|-- changelog.md         /* 它不在src下，但是值得提一下 */
```

components

最新的 Vue 官网也推出了 beta 版的风格指南，这里我以及整个开发团队的风格可能有些不一样。

如何样式超过 100 行，会单独分离出来便于维护

```
|-- components
|    |-- Recommend
|    |    |-- recommendItem.vue
|    |    |-- recommendList.vue
|    |-- LeaveMessageItem
|    |    |-- index.vue
|    |    |-- index.less
|    |-- WhiteSpace
|    |    |-- index.vue
```

filter

会在 main.js 中全部引入

```
import moment from 'moment'

// 处理文字过长的问题
```

```
const ellipsisText = (val, num) => {
 if (!val) return ''
 const len = val.toString().length
 return len > num ? val.substring(0, num) + '...' : val
}

// 格式化留言的日期
const isToday = (val) => {
 if (!val) return ''
 const today = moment().format('YYYY MM DD')
 const yesterday = moment().subtract(1, 'd').format('YYYY MM DD')
 const temp = moment(val).format('YYYY MM DD')
 if (today === temp) {
   return '今天 ' + moment(val).format('H:mm')
 } else if (yesterday === temp) {
   return '昨天 ' + moment(val).format('H:mm')
 } else {
   return moment(val).format('YYYY-MM-DD H:mm')
 }
}

// 返回当前时间（格式: 2018-01-09）
const currDate = () => moment().format('YYYY-MM-DD HH:mm:ss')

// ...
export {
 isToday,
 ellipsisText,
 currDate
 // ...
}
```

mixins

这里就举一个例子，项目中常常会有一些功能，又不便于写成组件，因为它并非实实在在的页面结构。我们可以抽成一个一个 mixin。比如在长列表，需要一个回到顶部的按钮，但是在下滑过程中我们需要将它隐藏，因为此刻用户用途并不想回到顶部。又比如播放器处于播放状态时，用户下滑时，应该隐藏或者切换成迷你状态。

```
const selfTouch = {
  data () {
    return {
      startx: '',
      starty: '',
      touchStatus: '未滑动',
      visible: false,
      timer: null
```

```
    }
  },
  watch: {
    touchStatus (val) {}
  },
  methods: {
    // 获取角度
    getAngle (angx, angy) {},

    // 根据起点终点返回方向 0-未滑动 1-向上 2-向下 3-向左 4-向右
    getDirection (startx, starty, endx, endy) {},

    // 滑动开始
    selfTouchstart () {},

    // 滑动结束
    selfTouchend () {},

    // 回到顶部
    toUp () {}
  }
}

export {
 selfTouch
}
```

views

这里先讲它，是因为 router 和 store 文件下的书写和命名都是基于它而来的

```
|-- views
|    |-- module1
|    |    |-- course-list
|    |    |-- course-detail
|    |    |-- course-leave-message
|    |    |-- course-leave-message-detail
|    |    |-- index.vue
|    |-- module2
|    |    |-- play-list
|    |    |-- task-list
|    |    |-- index.vue
|    |-- module3
|    |    |-- coupon
|    |    |-- history-record
|    |    |-- common-problem
|    |    |-- index.vue
```

文件夹命名都是小写，多单词采用‘-’链接，为了跟路由的path写法一致，每个文件对应的就是一个路由，只允许出现以 index.vue 命名的组件，less 规则和 component 是一致。

router

```
import Vue from 'vue'
import Router from 'vue-router'
// 懒加载
const _import = file => () => import('@/views/' + file + 'index.vue')

const Module1 = _import('module1')
const CourseList = _import('module1/course-list')
const CourseDetail = _import('module1/course-detail')
const CourseLeaveMessage = _import('module1/course-leave-message')
// ...

const Module2 = _import('module2')
// ...

const Module3 = _import('module3')
// ...

Vue.use(Router)
/**
 * path - 路由linkUrl
 * name - 路由名字
 * component - 对应的组件名
 * meta - 用来判断是否缓存整个路由组件
 * params - 路由路径的动态参数，尽量保证字段和后端定义的一致
 */
export default new Router({
  mode: 'history',
  routes: [
    {
      path: '/',
      redirect: '/module1',
      name: '模块1',
      component: Module1
    },
    {
      path: '/module1/course-list',
      name: '课程列表',
      component: CourseList,
      meta: {
        keepAlive: true
      }
```

```
    },
    {
      path: '/module1/course-leave-message/:courseId',
      name: '课程留言',
      component: CourseList
    },
})
```

在 main.js 也有 router 的相关代码，主要是为了实现产品的要求的功能

```
// 修改title的值
router.beforeEach((to, from, next) => {
 if (to.query.text) {
   // 一些动态的标题，比如：
   // '回复XXX'，'XXX人点过赞'，付费与免费跳转的路由不同却要保证标题一致
   document.querySelector('title').innerText = to.query.text
 } else {
   document.querySelector('title').innerText = to.name
 }
 next()
})

// 滚动到页面顶部
router.afterEach((to, from) => {
 if (to.name === '首页' || '推荐页' || '某某模块') window.scrollTo(0, 0)
})
```

未完待续...