小程序开发文档03/03

# 使用npm包

## 1、小程序对npm支持与限制

Npm包作用主要是提高小程序的开发效率，但有如下三个限制：

1. 不支持依赖node.js内置库的包
2. 不支持依赖于浏览器内置对象的包
3. 不支持依赖于C++插件的包

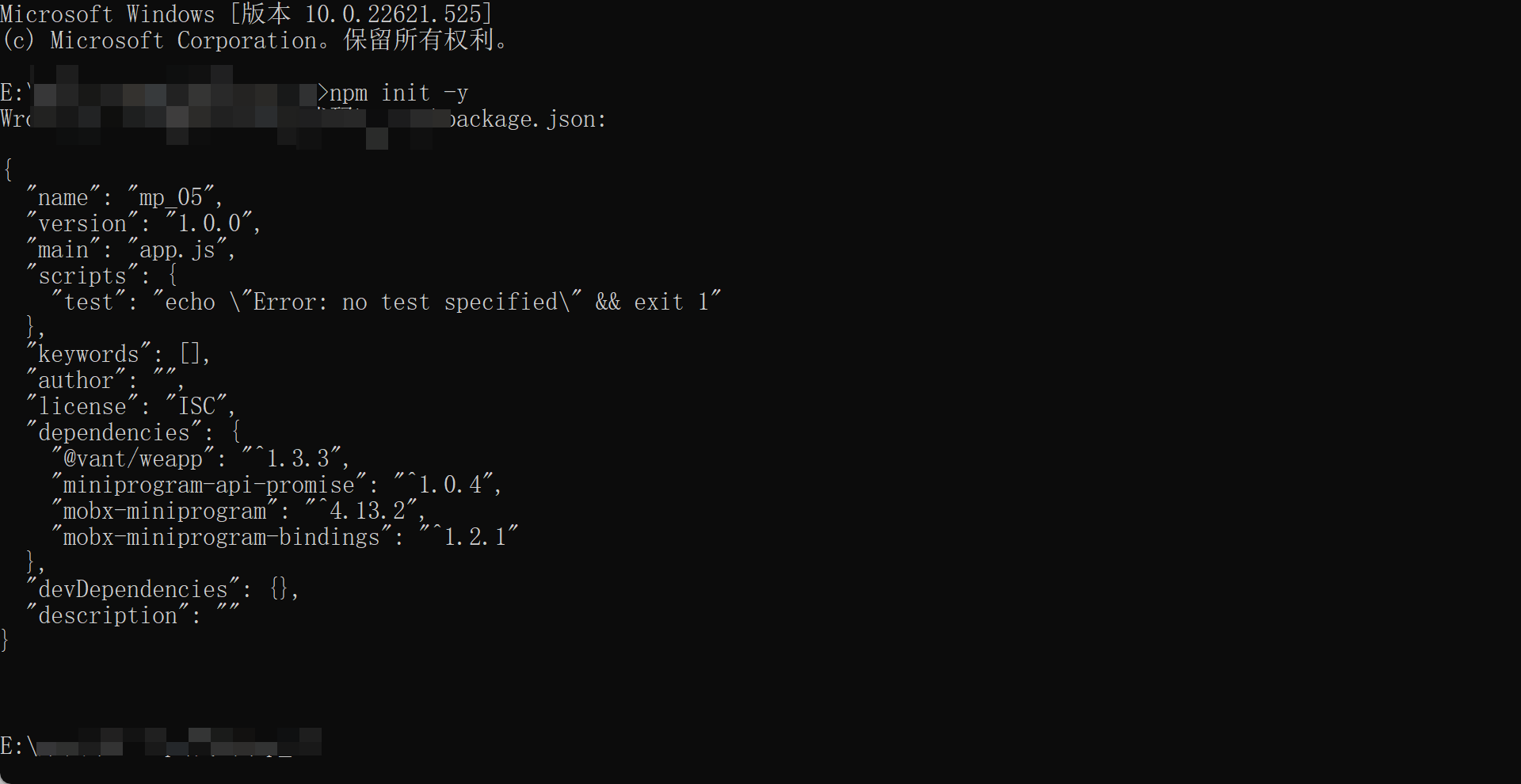
# 使用vant weapp

## 1、介绍：

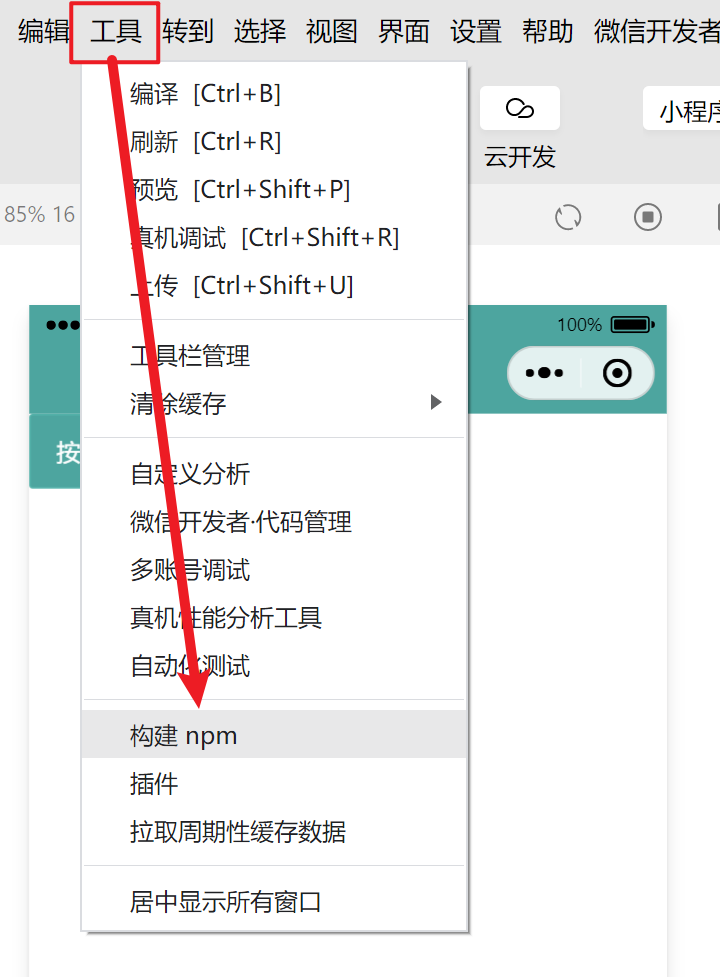
Vant Weapp是有赞前端团队开源的一套小程序UI组件库，助力开发者快速搭建小程序应用。使用的是MIT开源许可协议，对商业使用较友好。

## 2、安装Vant组件库

1. 通过npm安装



1. 构建npm包



1. 修改app.json

即去除：”style”:”v2”的代码。

## 3、使用Vant组件

安装好Vant组件库后，可以在app.json的usingComponents节点中引入需要的组件，即可在wxml中直接使用组件



## 4、定制全局主题样式

Vant Weapp使用CSS变量来实现定制主题

在app.wxss中，写入CSS变量，即可对全局生效



# 三、API Promise化

## 1、基于回调函数的异步API的缺点

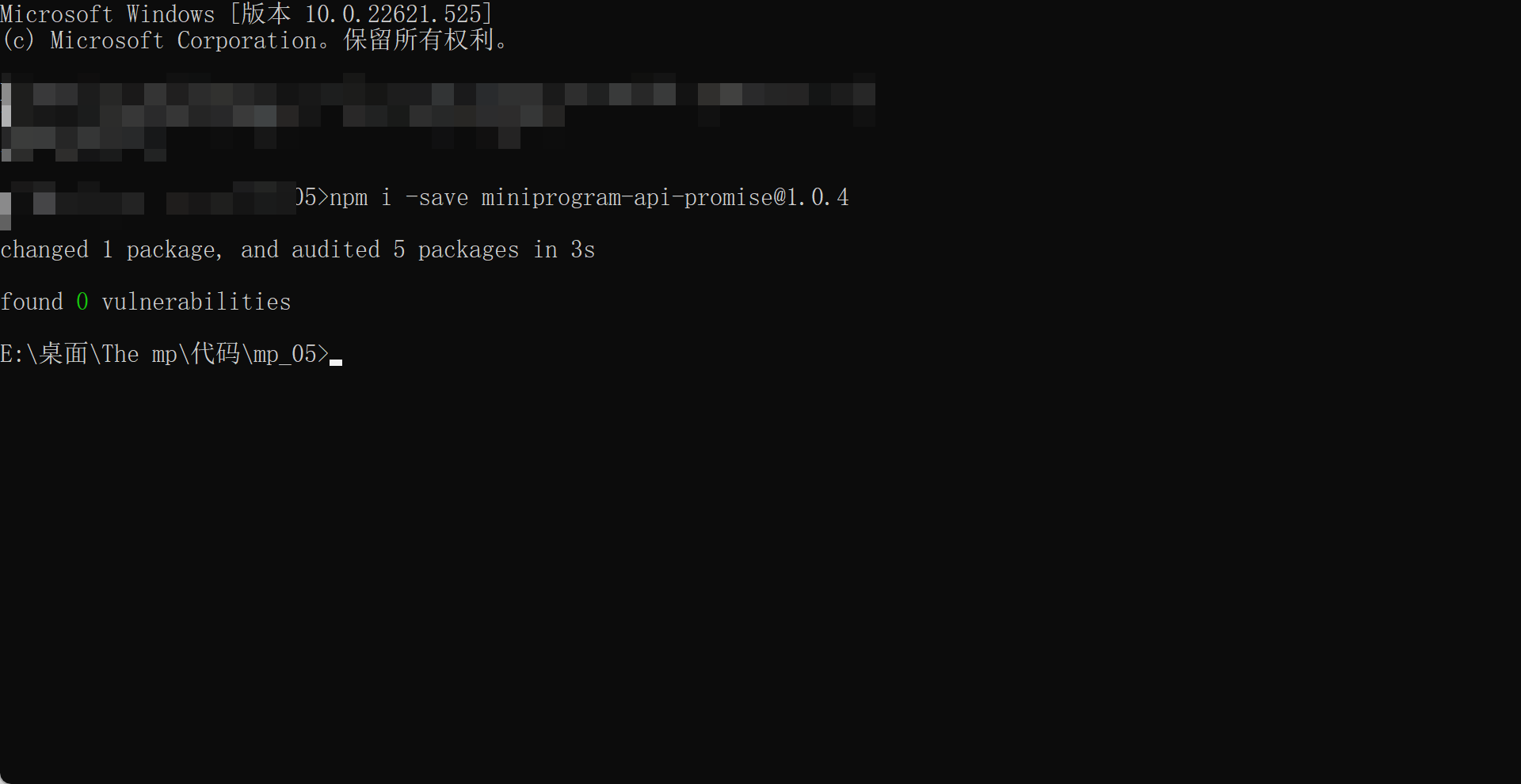
默认情况下，小程序官方提供的异步API都是基于回调函数实现，容易造成回调地狱的问题，代码的可读性、维护性差。

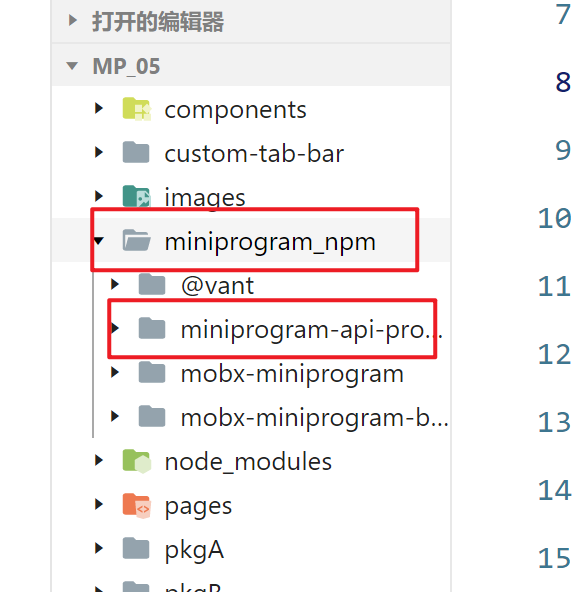
## 2、介绍

指的是通过额外的配置，将官方提供的、基于回调函数的异步API，升级改造为基于Promise的异步API，从而提高代码的可读性、维护性，避免回调地狱的问题。

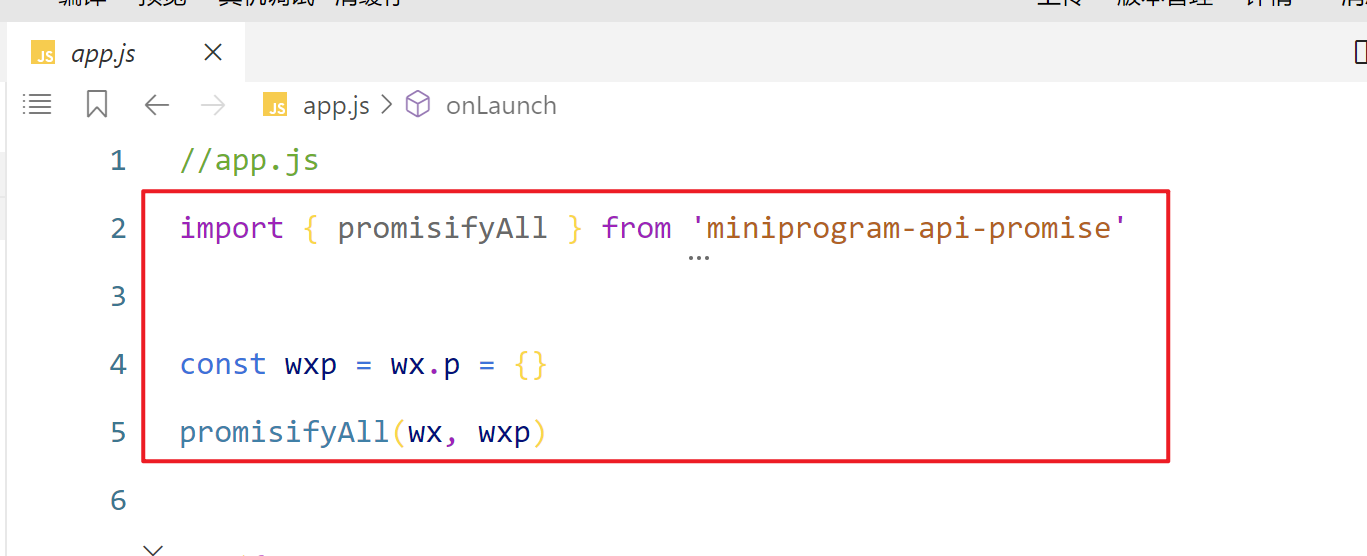
## 3、实现API Promise化

在小程序中，实现API Promise化主要依赖于miniprogram-api-promise这个第三方的npm包

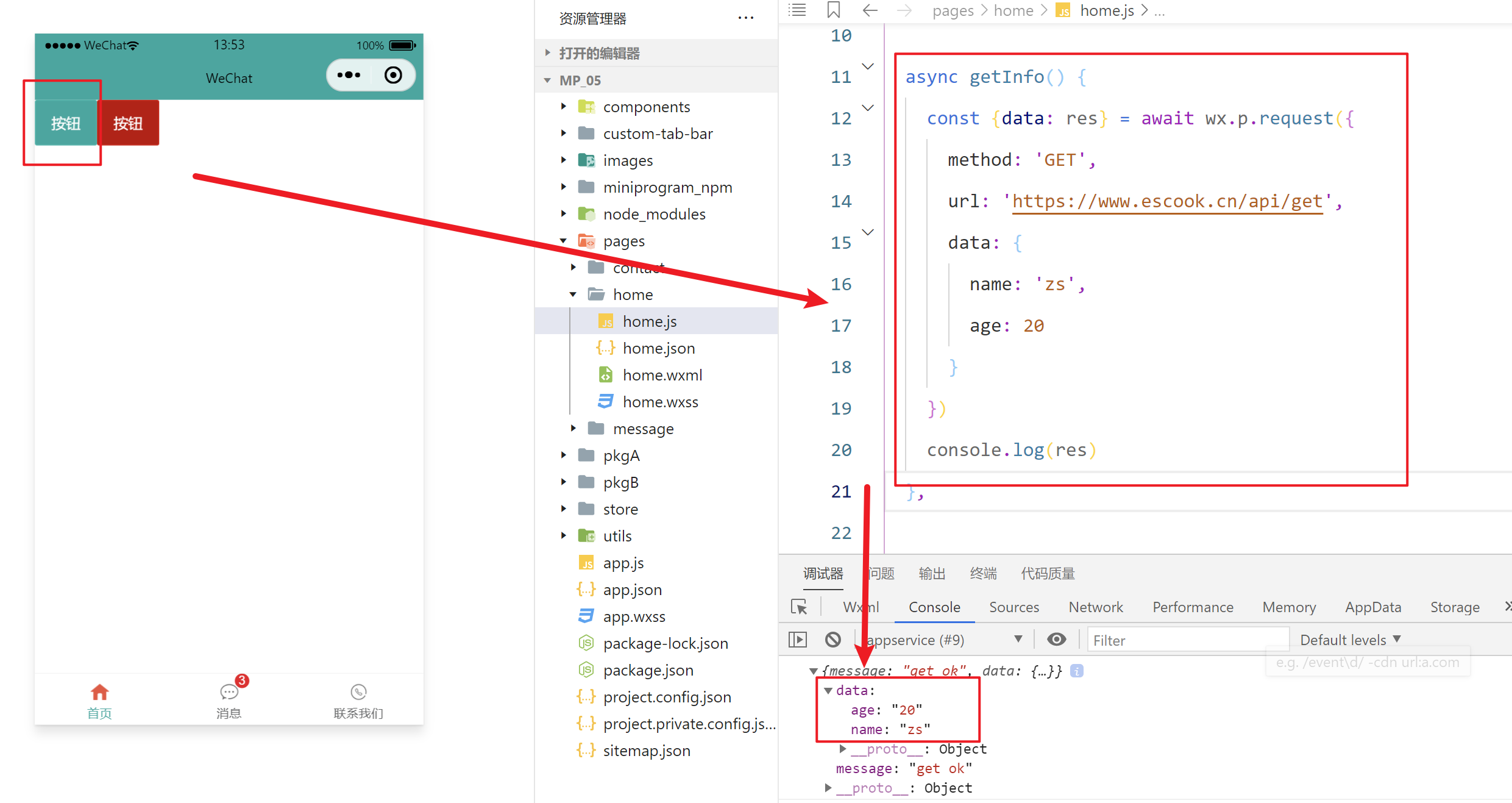




使用：



## 4、调用promise化之后的异步API



# 四、全局数据共享

## 1、介绍：

又叫状态管理，是为了解决组件之间数据共享的问题。

常用全局数据共享方案：Vuex、Redux、MobX等

## 2、小程序的全局数据共享方案

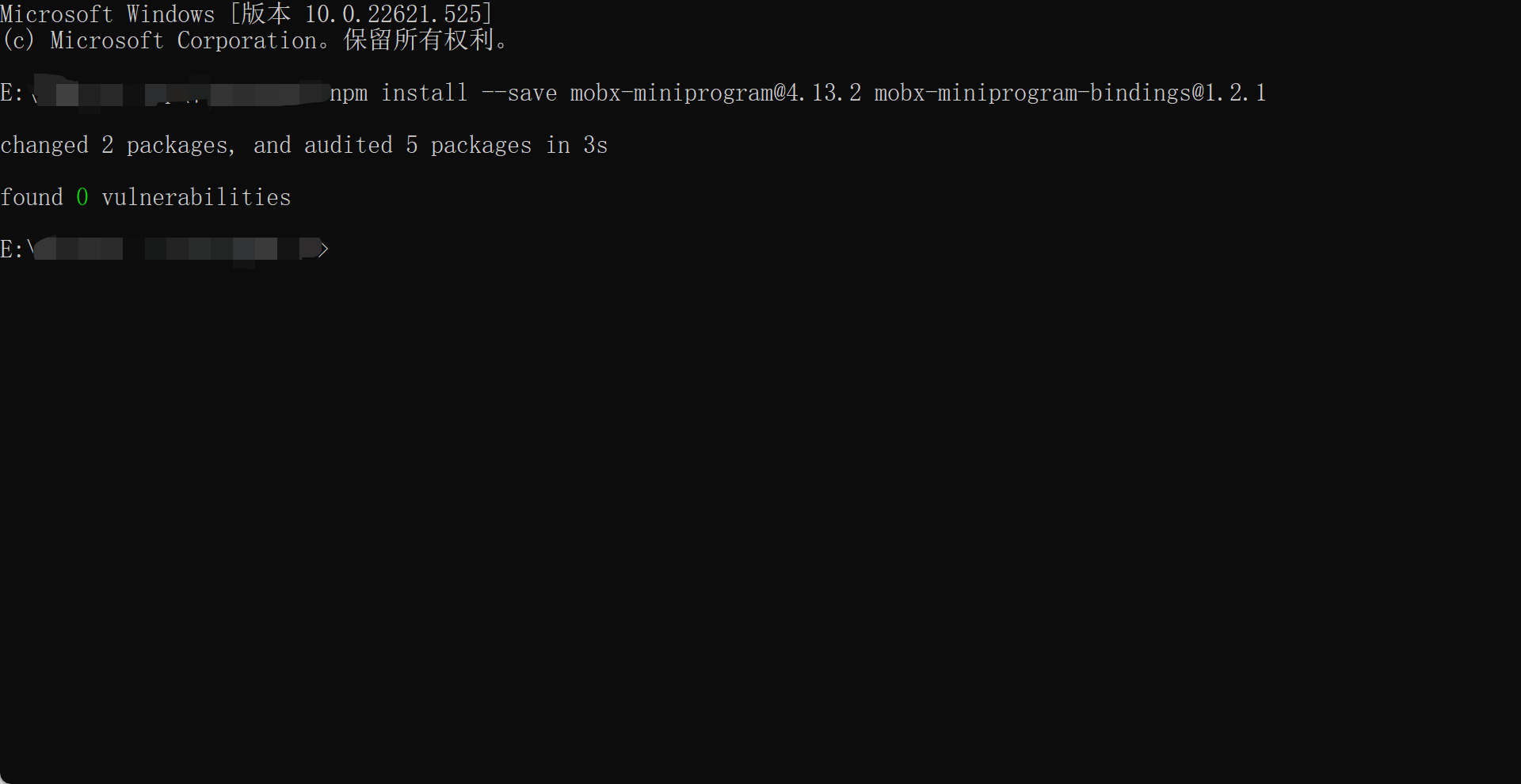
在小程序中，可使用mobx-miniprogram配合mobx-miniprogram-bindings实现全局数据共享。其中：

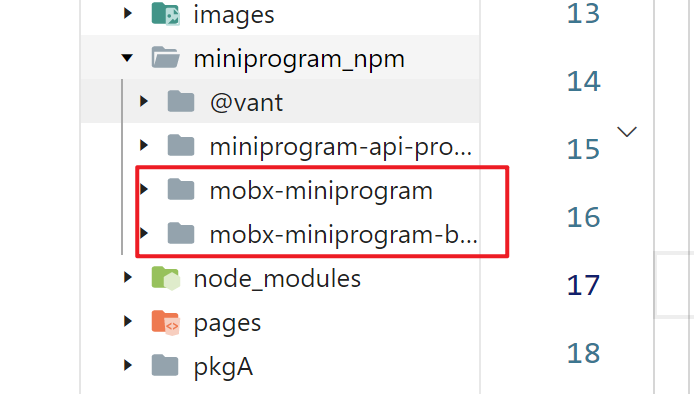
mobx-miniprogram用来创建Store实例对象

mobx-miniprogram-bings用来把Store中的共享数据或方法，绑定到组件或页面中使用

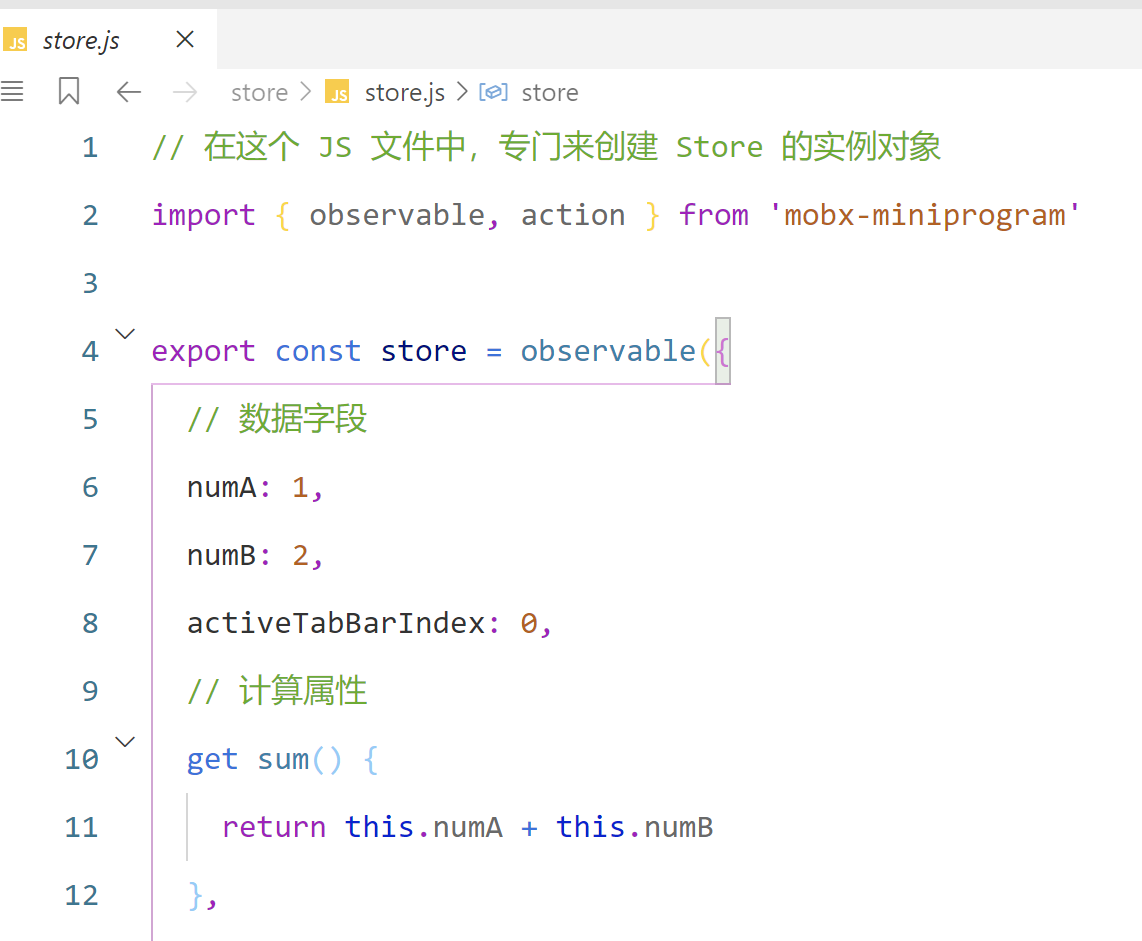
# 五、MobX

## 1、安装

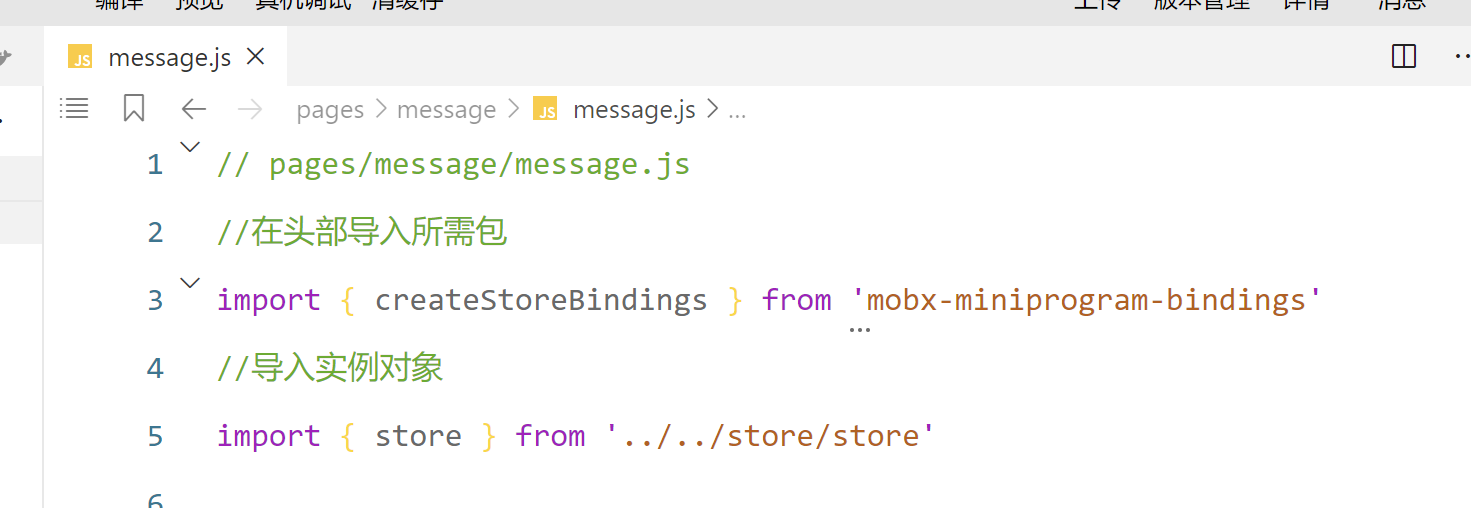




## 2、创建MobX的store实例



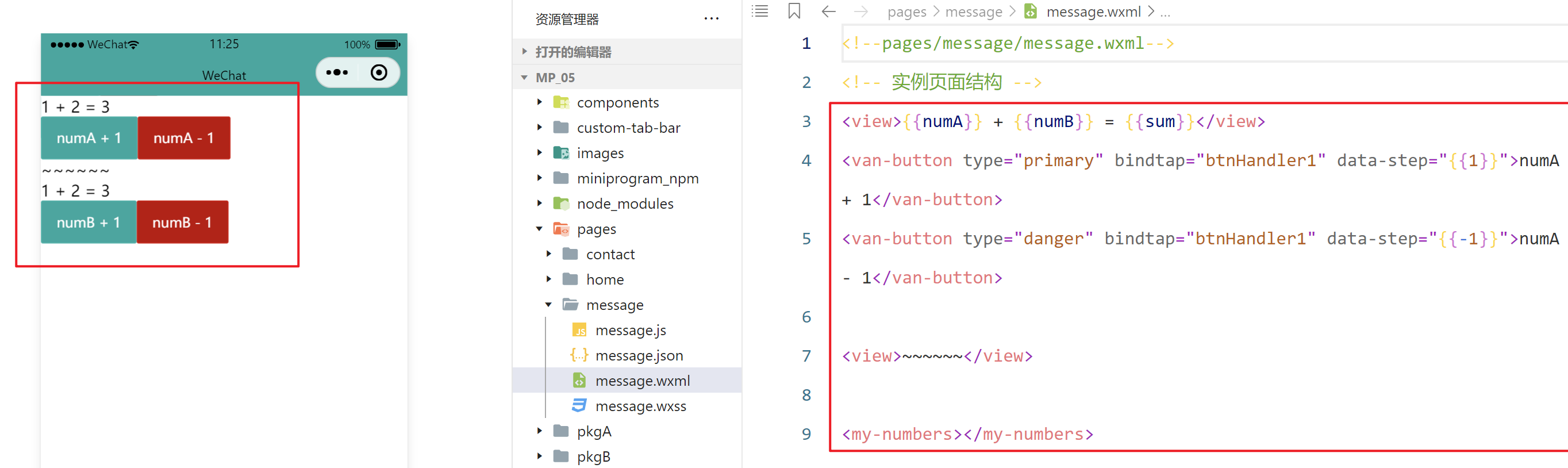
## 3、将store中的成员绑定到页面中





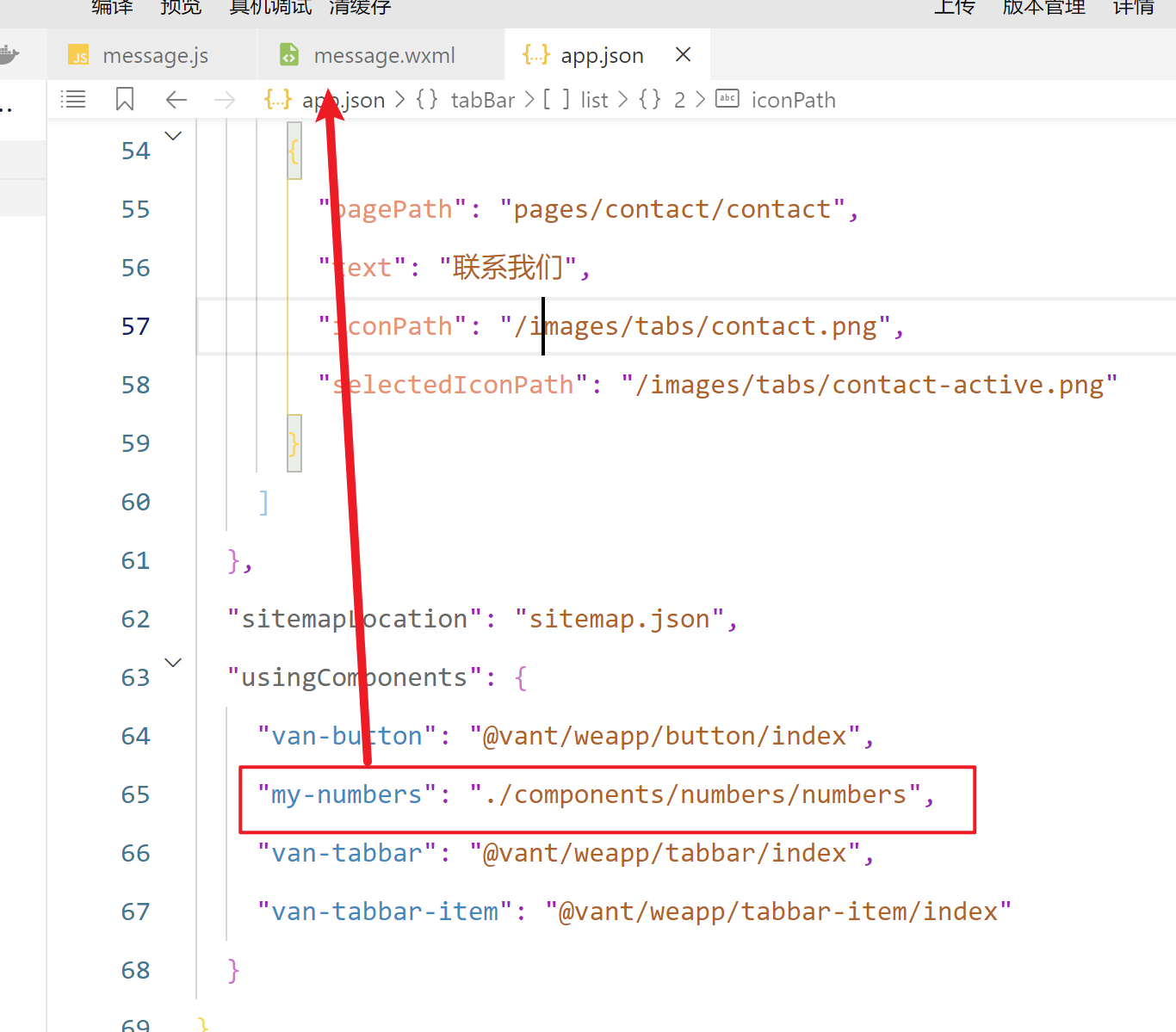


## 4、在页面上使用Store中的成员





## 5、将Store成员绑定到组件中





# 六、分包

## 1、介绍：

指把一个完整的小程序项目，按照需求划分为不同的子包，在构建时打包成不同的分包，用户在使用适按需进行加载。

## 2、好处

1）优化小程序首次启动的下载时间

2）在多团队共同开发时可以更好的解耦协作

## 3、分包前项目的构成

分包前，小程序中所有的页面和资源都打包在一起，导致整个项目体积过大，影响小程序首次启动的下载时间。

小程序项目

等

WXS

JS脚本

其他页面

TabBar

公共资源

所有页面

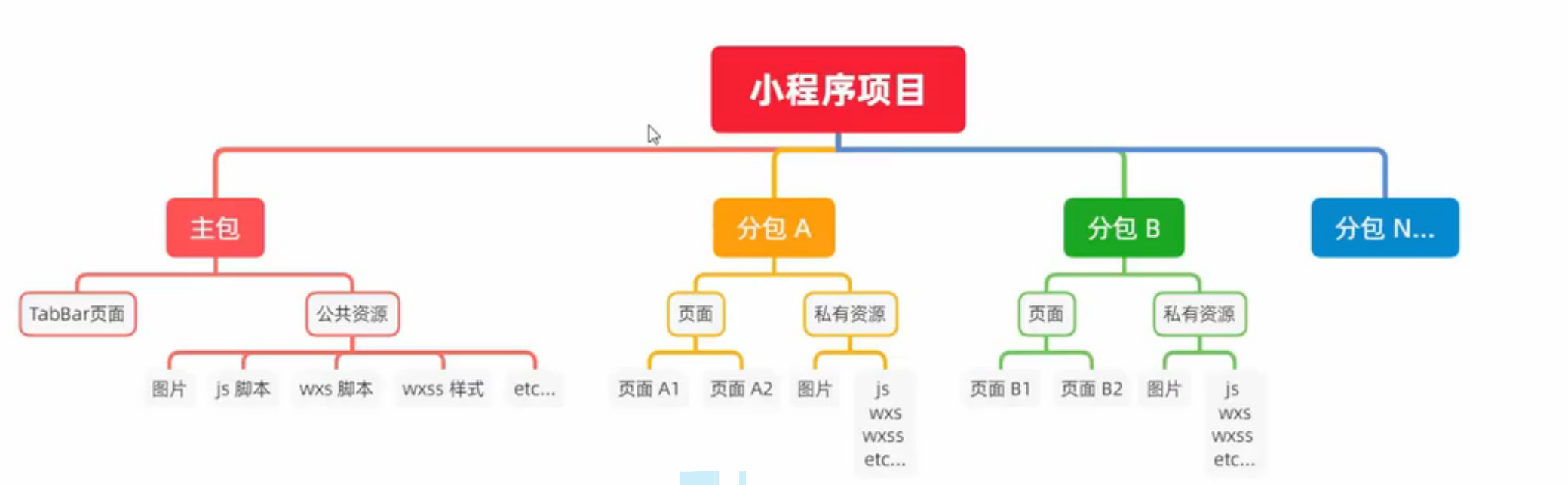
图片

## 4、分包后项目的构成

分包后，小程序有1个主包和多个分包组成：

主包：一般只包含项目的启动页面或TabBar也米娜、以及所有分包都需要用到的一些公共资源。

分包：只包含和当前分包有关的页面和私有资源。



## 5、分包的加载规则

### 5.1、在小程序启动时，默认会下载主包并启动主包内页面

在tabBar页面需要放到主包内

### 5.2、当用户进入分包内某个页面时，客户端会把对应分包下载下拉，下载完成后再进行展示

非tabBar页面可以按照功能的不同，划分为不同的分包之后，进行按需下载

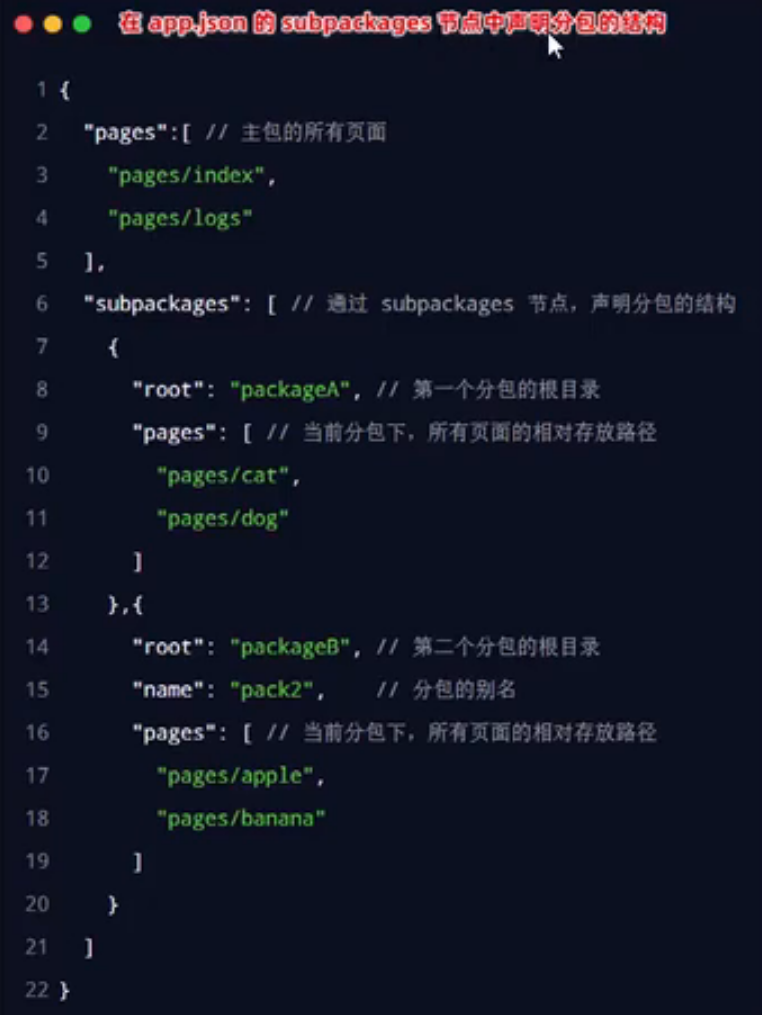
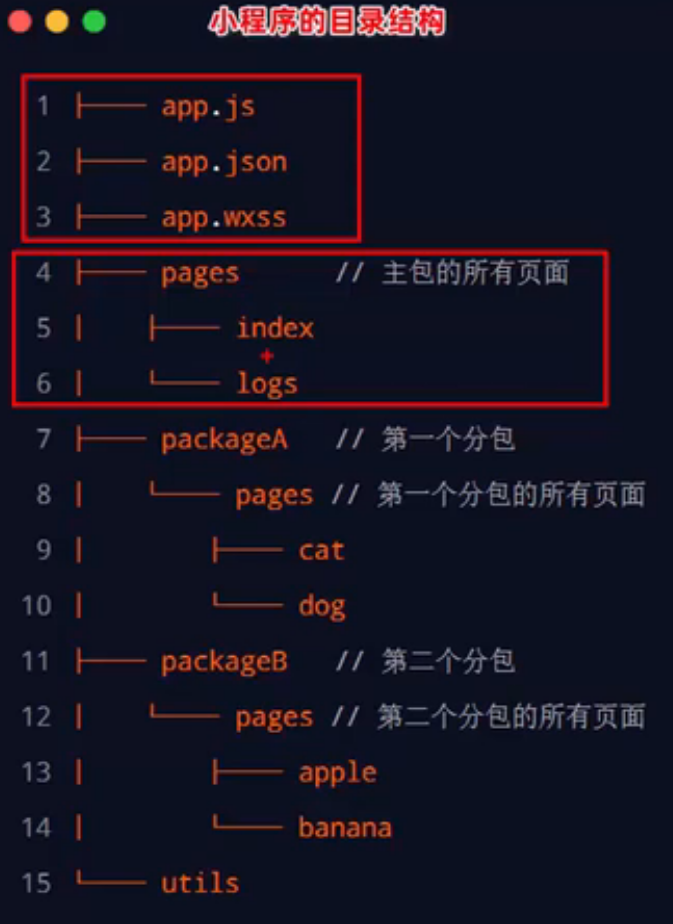
## 6、分包的体积限制

1）整个小程序所有分包大小不超过16M（主包+所有分包）

2）单个分包/主包大小不能超过2M

# 七、使用分包

## 1、配置方法



## 2、打包原则

1）小程序会按subpackages的配置进行分包，subpackages之外的目录将被打包到主包中

2）主包也可以有自己的pages（即最外层的pages字段）

3）tabBar页面必须在主包内

4）分包之间不能互相嵌套

## 3、引用原则

1）主包无法引用分包的私有资源

2）分包之间不能相互引用私有资源

3）分包可以引用主包内的公共资源

# 八、独立分包

## 1、介绍：

独立分包本质上也是分包，但特殊在可以独立于主包和其他分包而单独运行

## 2、区别：

最主要区别：是否依赖于主包才能运行

普通分包必须依赖于主包才能运行

独立分包可以在不下载主包的情况下，独立运行

## 3、应用场景

开发者可以按需，将某些具有一定功能独立性的页面配置到独立分包中。

原因：

1）当小程序从普通分包页面启动时，需要首先下载主包

2）而独立分包不依赖主包即可运行，可以很大程度提升分包页面的启动速度

## 4、配置方法



## 5、引用原则

1）独立分包和普通分包以及主包之间，是相互隔绝的，不能相互引用彼此的资源。

2）主包无法引用独立分包内的私有资源

3）独立分包之间，不能相互引用私有资源

4）独立分包和普通分包之间，不能相互引用私有资源

5）特别注意：独立分包中不能引用主包内的公共资源（而普通分包可以）

# 九、分包预下载

## 1、介绍：

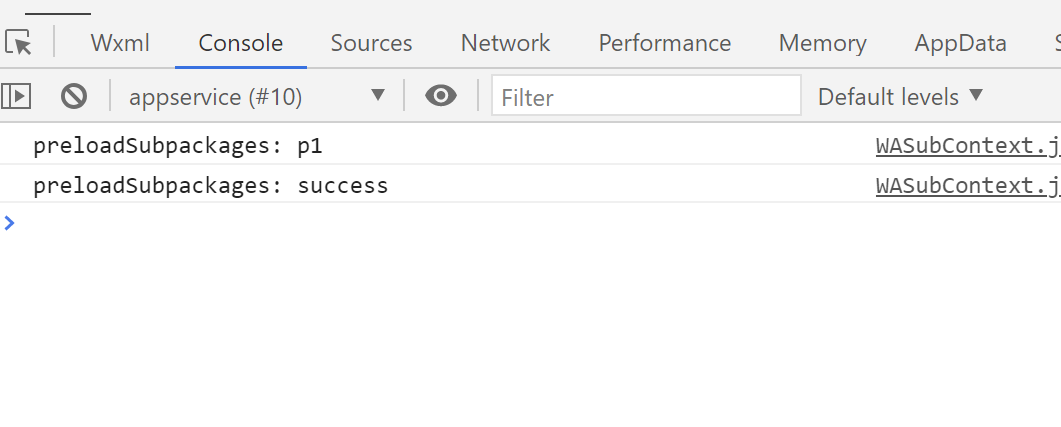
在进入小程序的某个页面时，由框架自动预下载可能需要的分包，从而提升进入后续分包页面的启动速度。

## 2、配置分包预下载

会在进入指定的页面时触发。在app.json中，使用preloadRule节点定义分包的预下载规则



预下载成功：



## 3、分包预下载的限制

同一个分包中的页面享有共同预下载的大小限额为2M