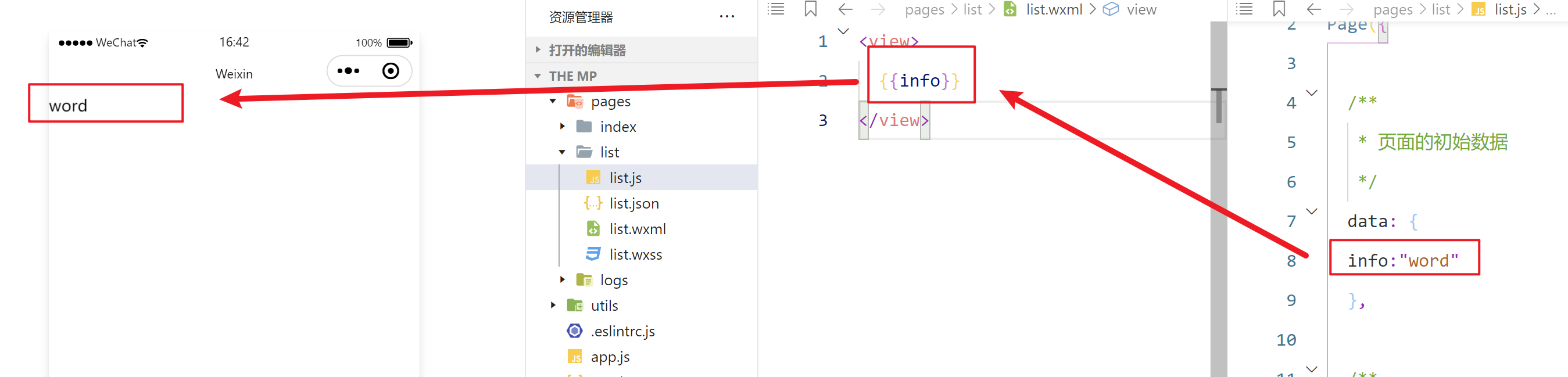
小程序开发文档02/06

# 一、WXML模板语法

## 1、 数据绑定

数据绑定的基本原则：  
   在data中定义数据  
   在WXML中使用数据

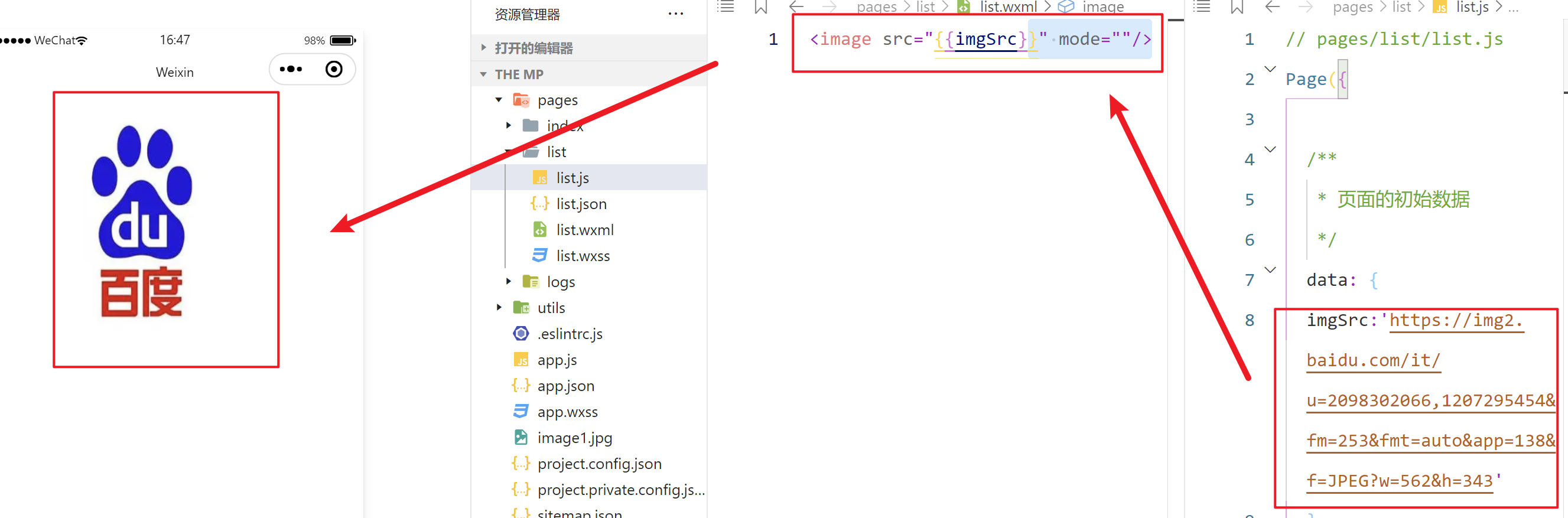
在data中定义页面的数据：  
    在页面对应的.js文件中，把数据定义到data对象中即可



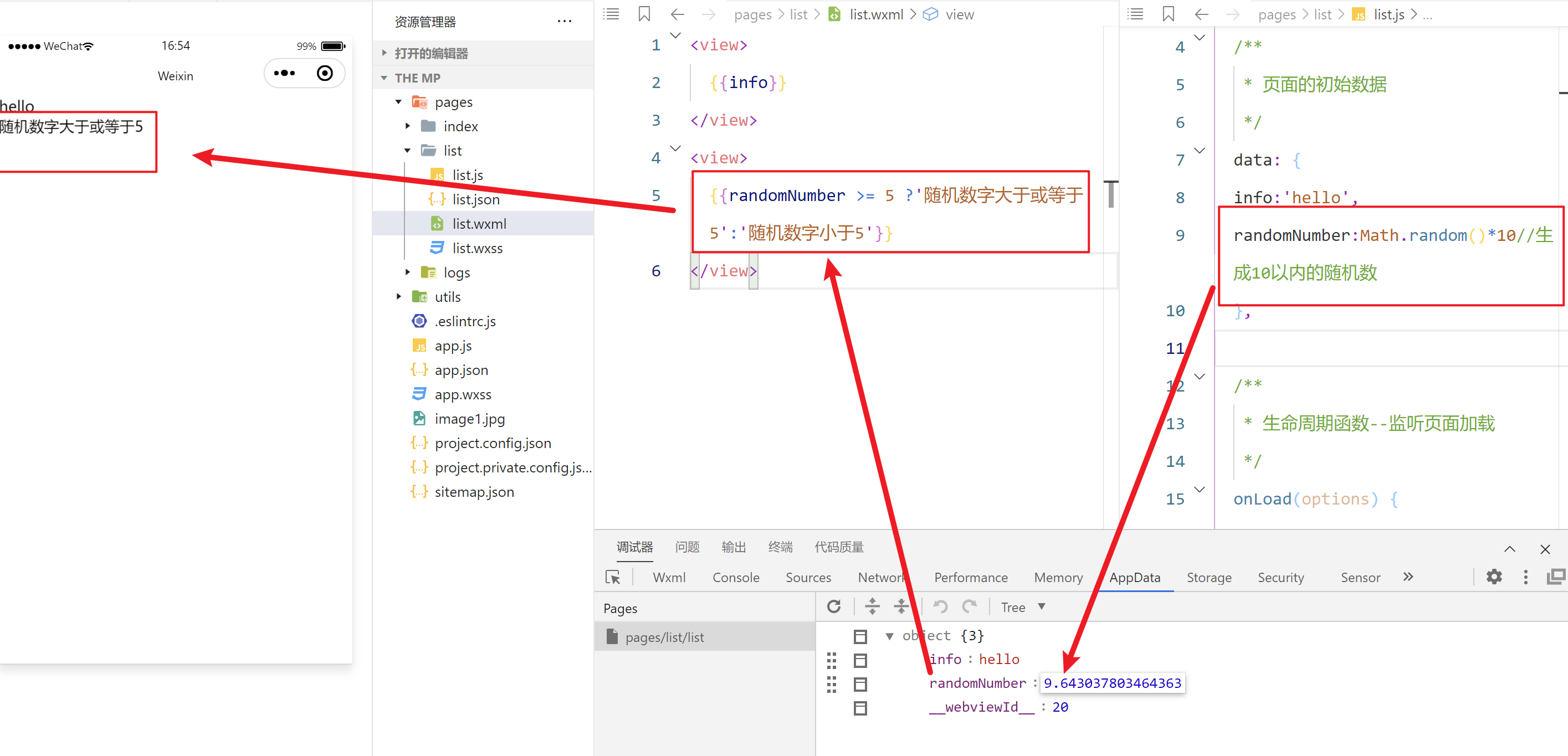
## 2、Mustache语法应用场景

### 2.1主要应用场景：

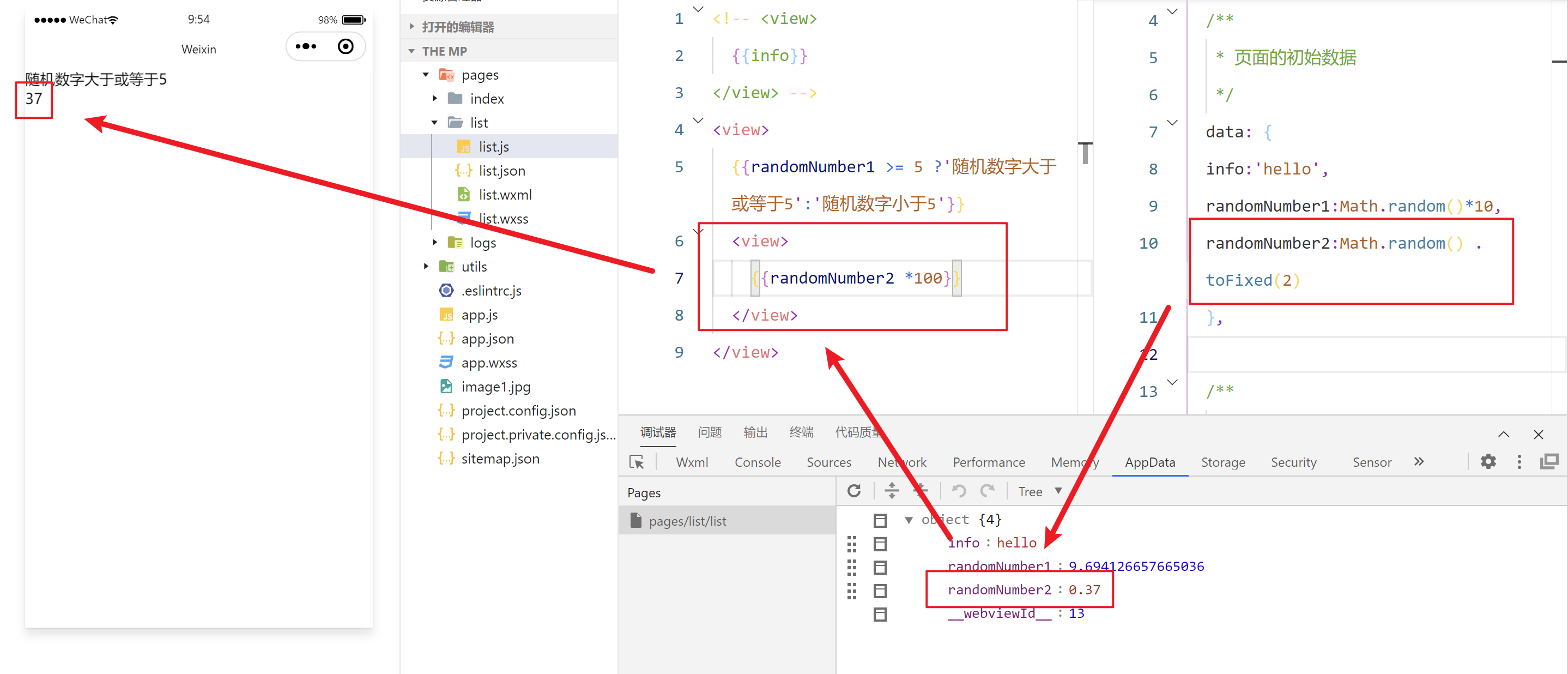
绑定内容、绑定属性、运算（三元运算、算术运算等）。



### 2.2三元运算



### 2.3算数运算



## 3、事件绑定

### 3.1什么是事件：

事件是渲染层到逻辑层的通讯方式。通过事件可以将用户在渲染层产生的行为，反馈到逻辑层进行业务的处理。

### 3.2小程序常用事件

|  |  |  |
| --- | --- | --- |
| 类型 | 绑定方式 | 事件描述 |
| Tap | Bindtap或bind:tap | 手指触摸后马上离开，类似HTML中的click事件 |
| Input | bindinput或bind:input | 文本框的输入事件 |
| Change | bindchange或bind:change | 状态改变时触发 |

### 3.3事件对象的回调列表

当事件回调触发的时候，会收到一个事件对象event，它的详细属性如下表：

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 说明 |
| type | String | 事件类型 |
| timeStamp | Integer | 页面打开到触发事件所经过的毫秒数 |
| target | Object | 触发事件的组件的一些属性值集合 |
| currentTarget | Object | 当前组件的一些属性值集合 |
| detrail | Object | 额外的信息 |
| touches | Arrary | 触发事件，当前停留在屏幕中的触摸点信息的数组 |
| changedTouches | Array | 触摸事件，当前变化的触摸点信息的数组 |

### 3.4target和currentTarget的区别

Target是触发该事件的源头组件，而currenttarget则是当前事件绑定的组件。

点击内部的按钮时，点击事件以冒泡的方式向外扩散，也会触发外层的view的tap事件处理函数。

此时对外层的view来说：

1、e.target指向的是触发事件的源头组件，因此 e.target是内部的按钮组件；

2、e.currentTarget指向的是当前正在触发事件的那个组件，因此，e.currentTarget是当前的view组件。

### 3.5bindtap的语法格式

在小程序中，不存在Html中的onclick鼠标点击事件，而是通过tap事件来响应用户的触摸行为。

1、通过bindtap，可以为组件绑定tap触摸事件

使用步骤：

步骤1:通过bindtap，可以为组件绑定tap触摸事件，语法如下：

<button type="primary" bindtap="btnTapHandler">按钮</button>

步骤2:在页面的.js文件中定义对应的事件处理函数，事件参数通过形参event（一般写出e）来接受：

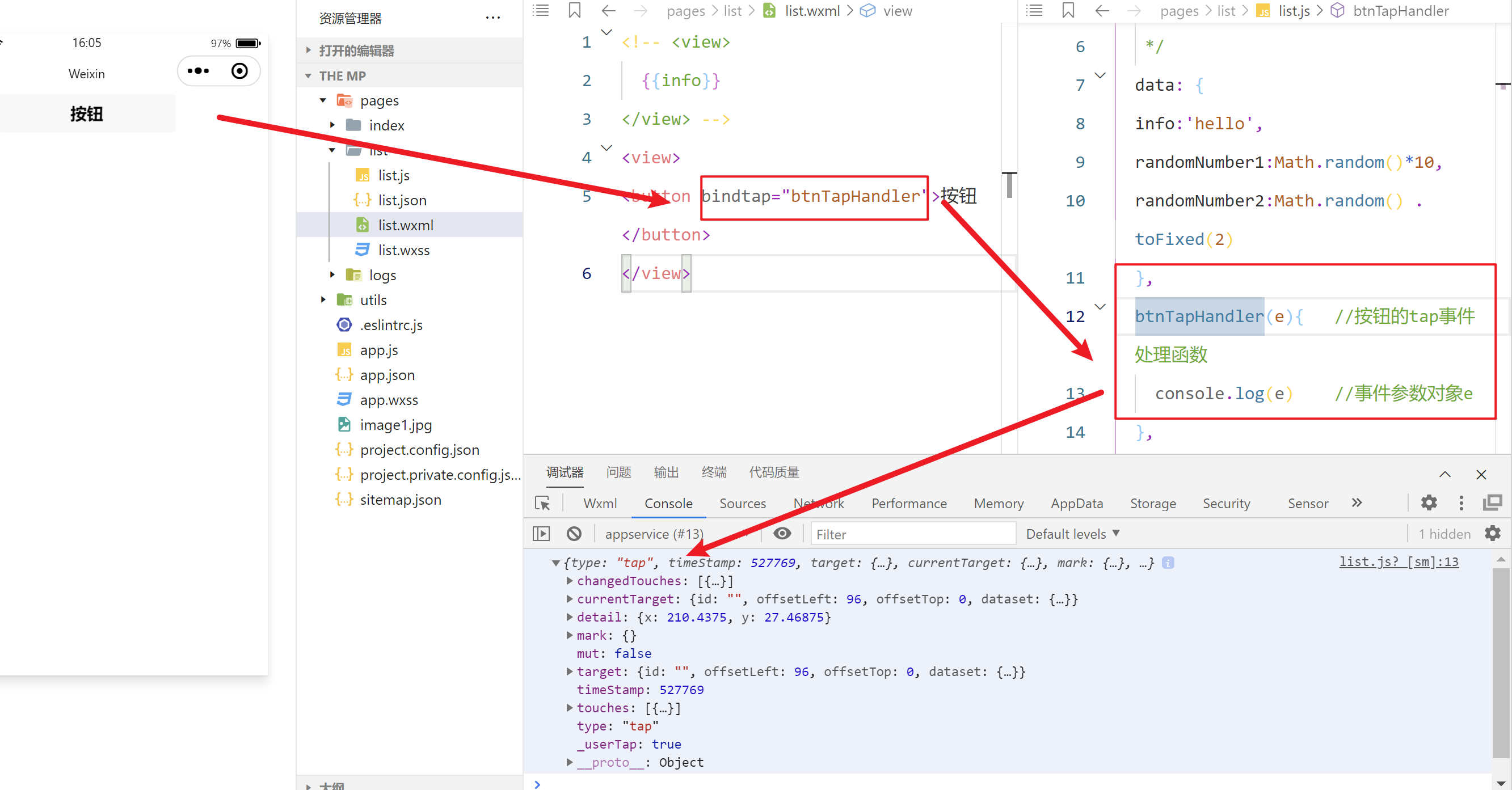
Page({

btnTapHandler(e){ //按钮的tap事件处理函数

console.log(e) //事件参数对象e

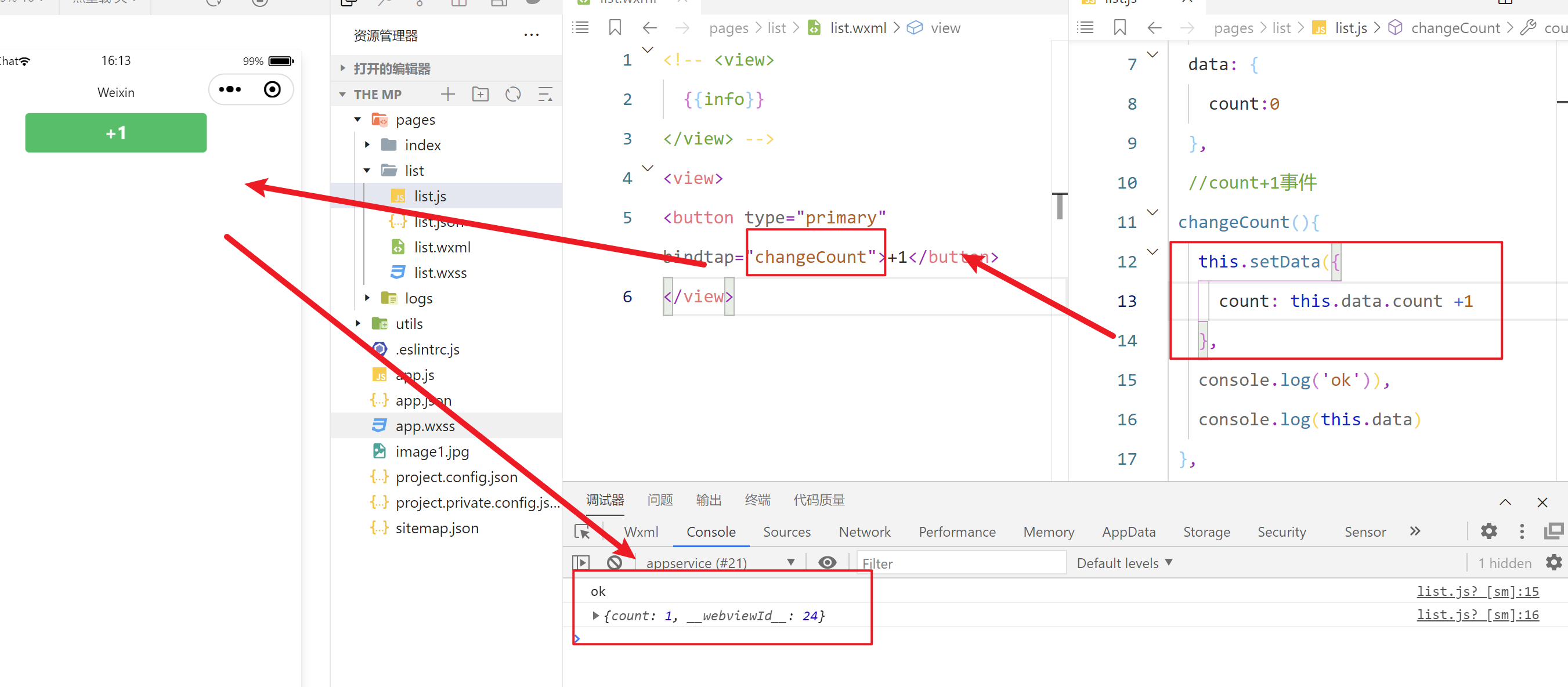
}

})



### 3.6 在事件处理函数中为data中的数据赋值

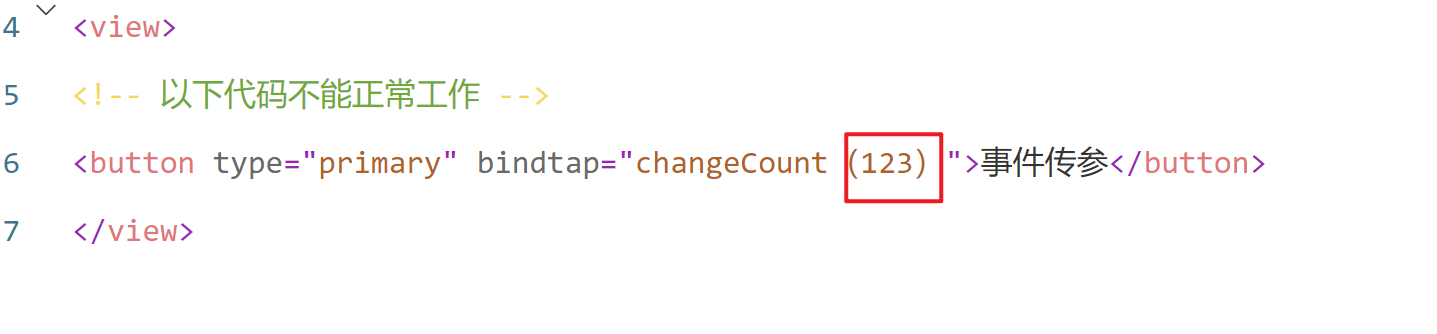
通过调用this.setData(dataObject)方法，可以给页面data中的数据重新赋值



## 4、事件传参

### 4.1小程序的事件传参

小程序传参不能再绑定事件的同时为事件处理函数传递参数。



可以为组件提供“data-\*”自定义属性传参，其中\*代表的是参数的名字



1、info会被解析为参数的名字，数值2会被解析为参数的值。

2、在事件处理的函数中，通过event.target.dataset.参数名 即可获得到具体参数的值

### 4.2bindinput的语法格式

在小程序中，通过input事件来响应文本框的输入事件，语法如下：

步骤1-通过bindinput，可以为文本框绑定输入事件

<input bindinput="inputHandler" />

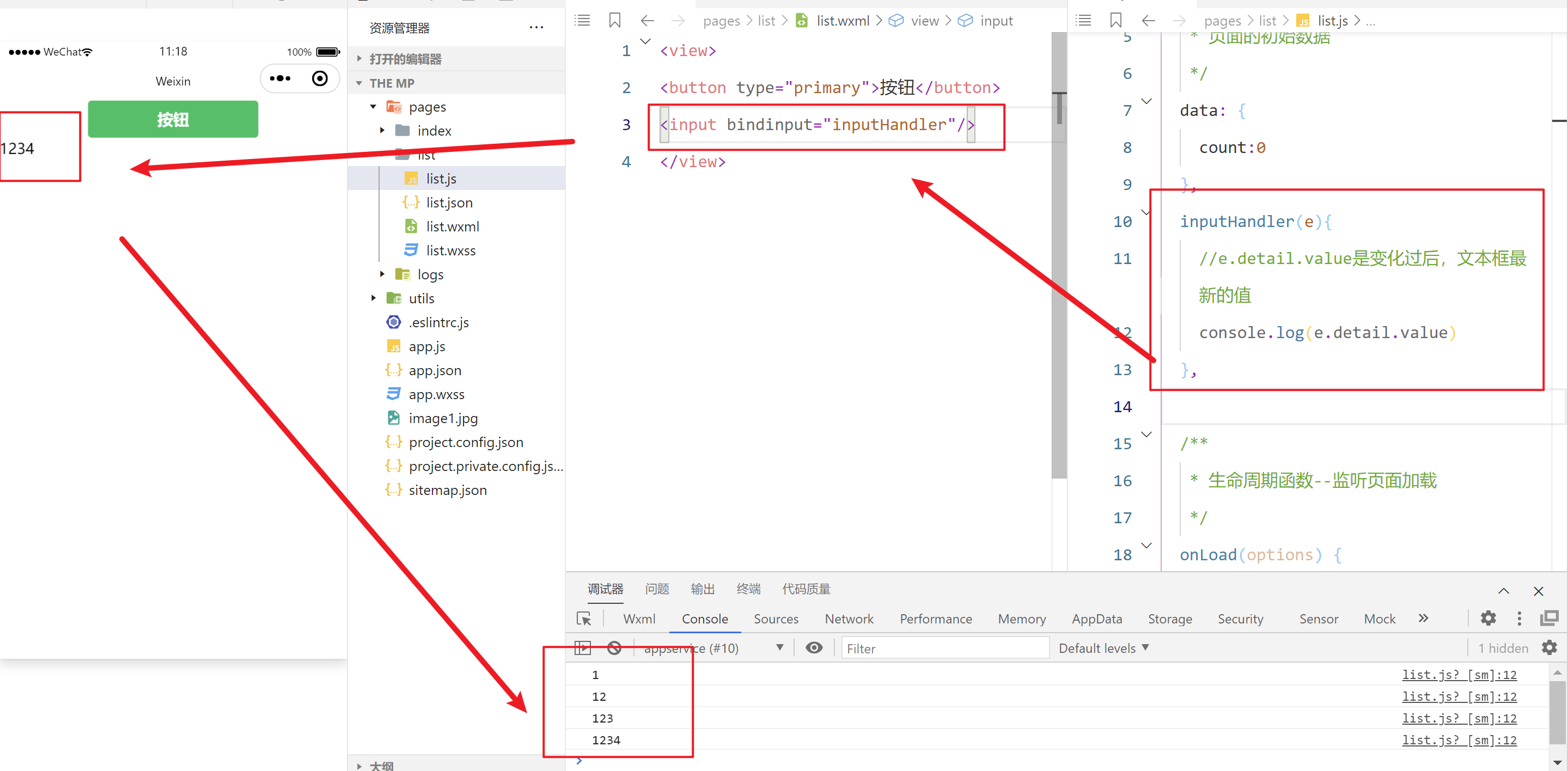
步骤2-在页面的.js文件中定义事件的处理函数：

inputHandler(e){

//e.detail.value是变化过后，文本框最新的值

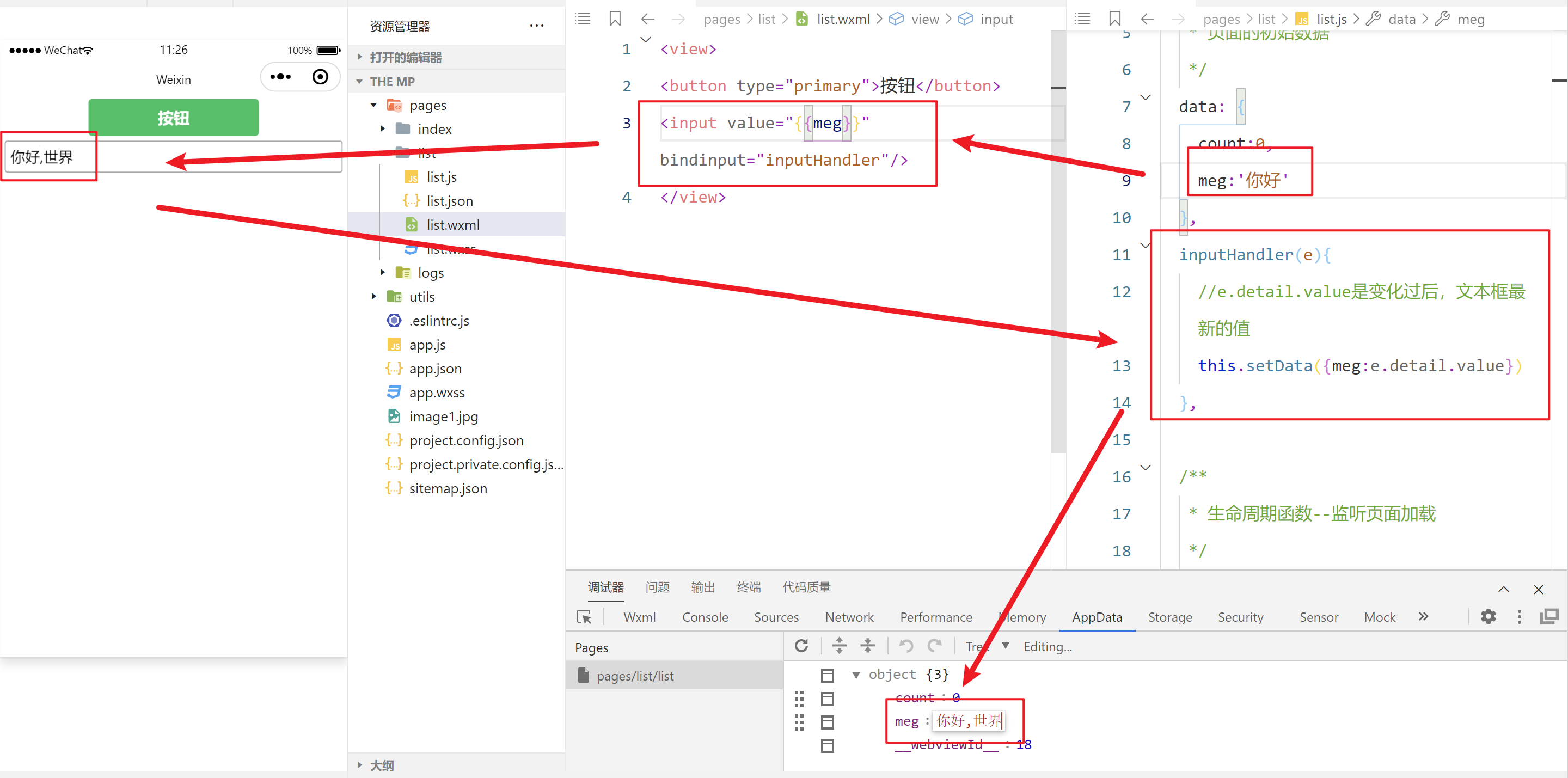
consloe.log(e.detail.value)

}



### 4.3 实现文本框和data之间的数据同步

实现步骤：1）定义数据 2）渲染结构 3）美化样式 4）绑定input事件处理函数



## 5、条件渲染

### 5.1wx:if

wx:if用来来判断是否需要渲染该代码块

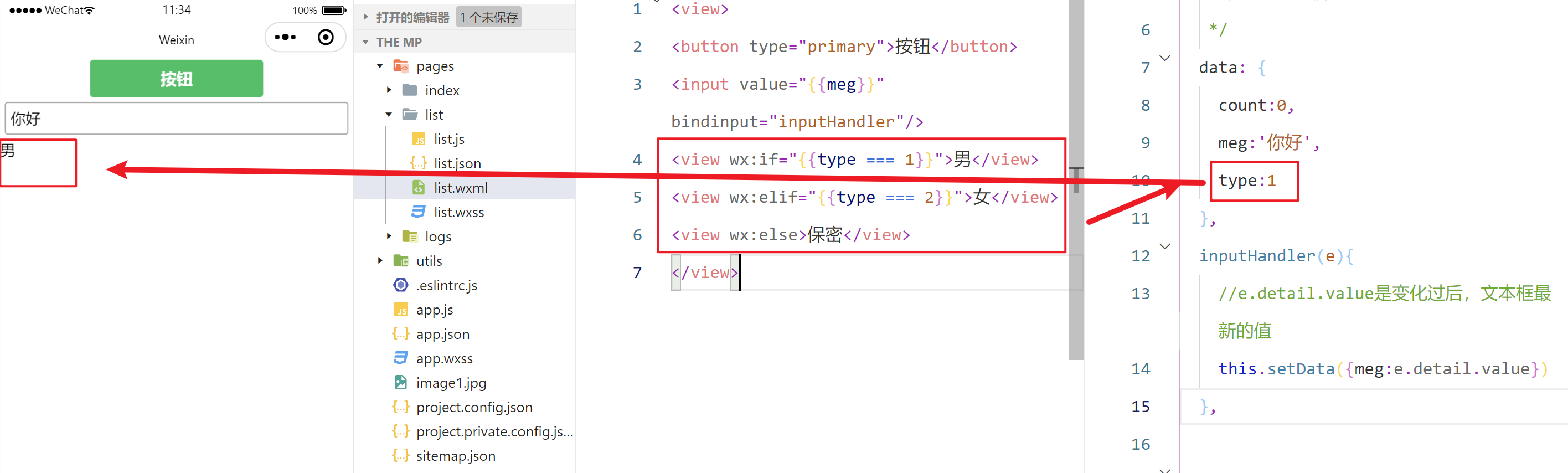
<view wx:if="{{condition}}">True</view>

也可以用wx:elif和wx:else来添加else判断：

<view wx:if="{{type === 1}}">男</view>

<view wx:elif="{{type === 2}}">女</view>

<view wx:else>保密</view>



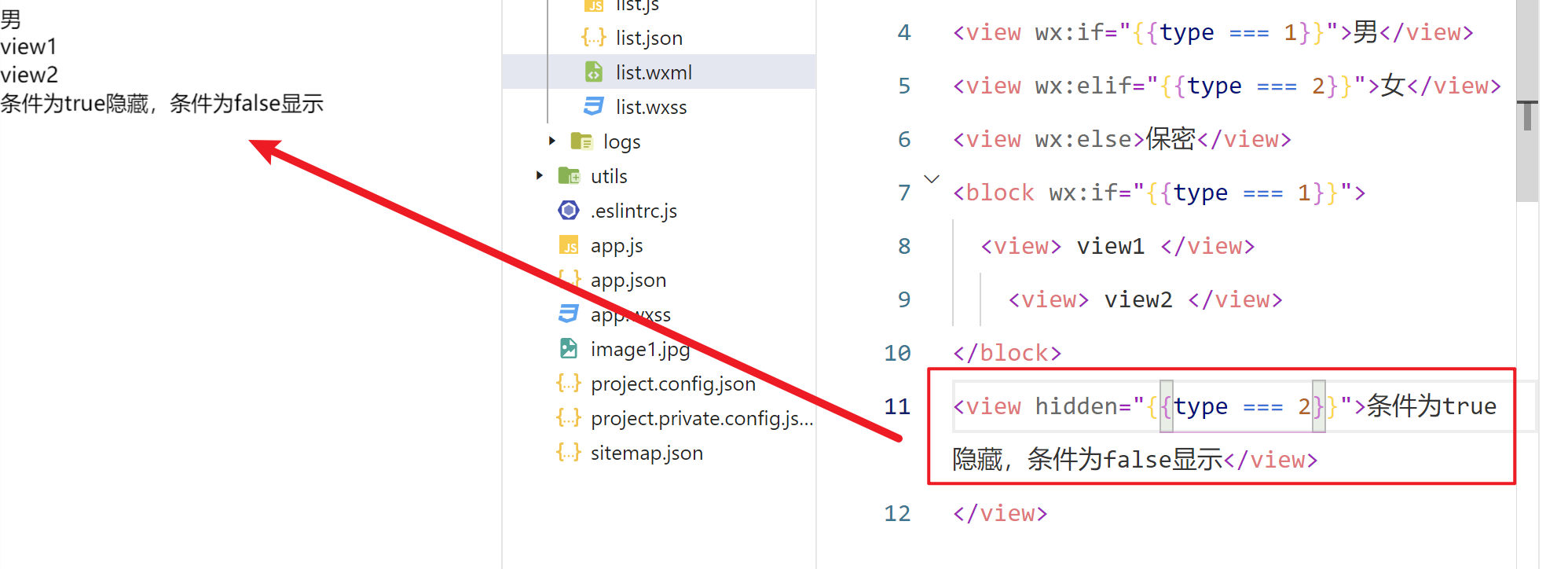
结合<block>使用wx:if

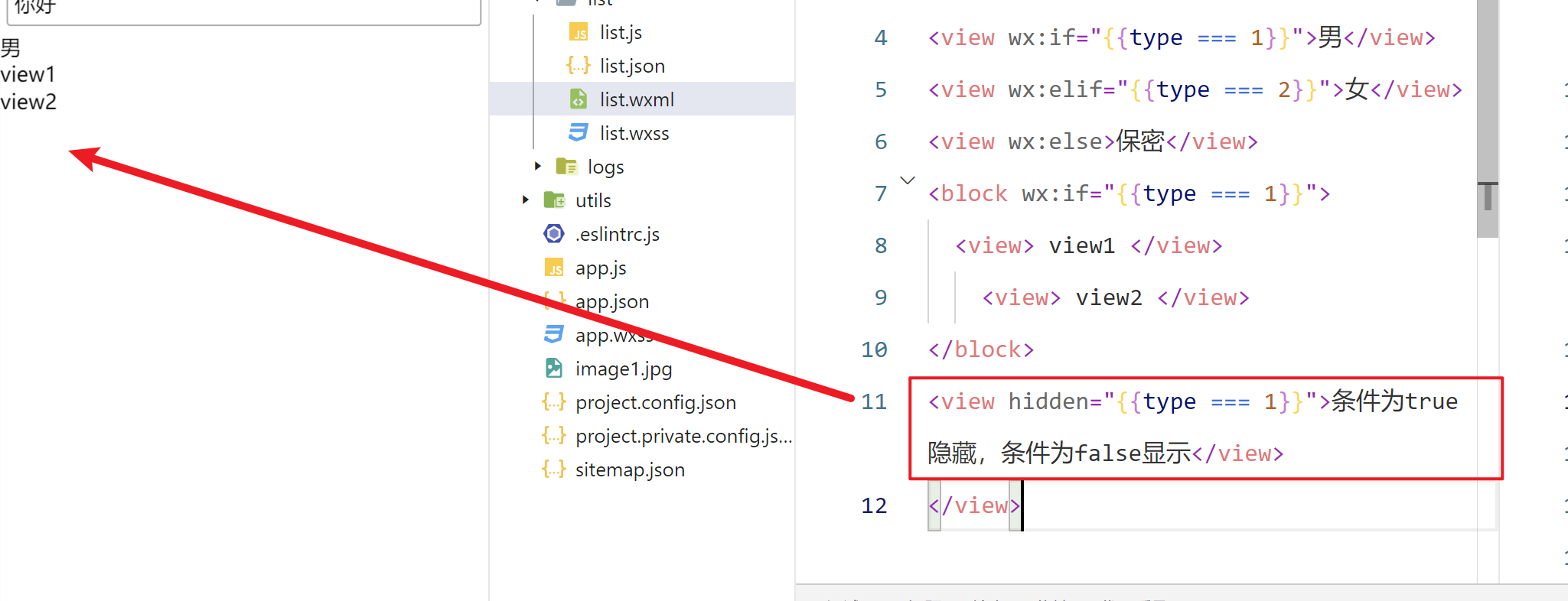
如果要一次性控制多个组件的展示与隐藏，可以使用一个<block></block>标签将多个组件包装起来，并在<block>标签上使用wx:if控制属性



注意：<block>并不是一个组件，它只是一个包裹性质的容器，不会再页面中做任何渲染，比使用<view>节省了不必要的节点。

 Hidden：在小程序中，直接使用hidden=“{{condition}}”也能控制元素的显示与隐藏：





### 5.2wx:if 与 hidden的对比

1）运行方式不同

前者以动态创建和移除元素的方式，控制元素的展示与隐藏

后者以切换样式的方式，控制元素的显示和隐藏

2）使用建议

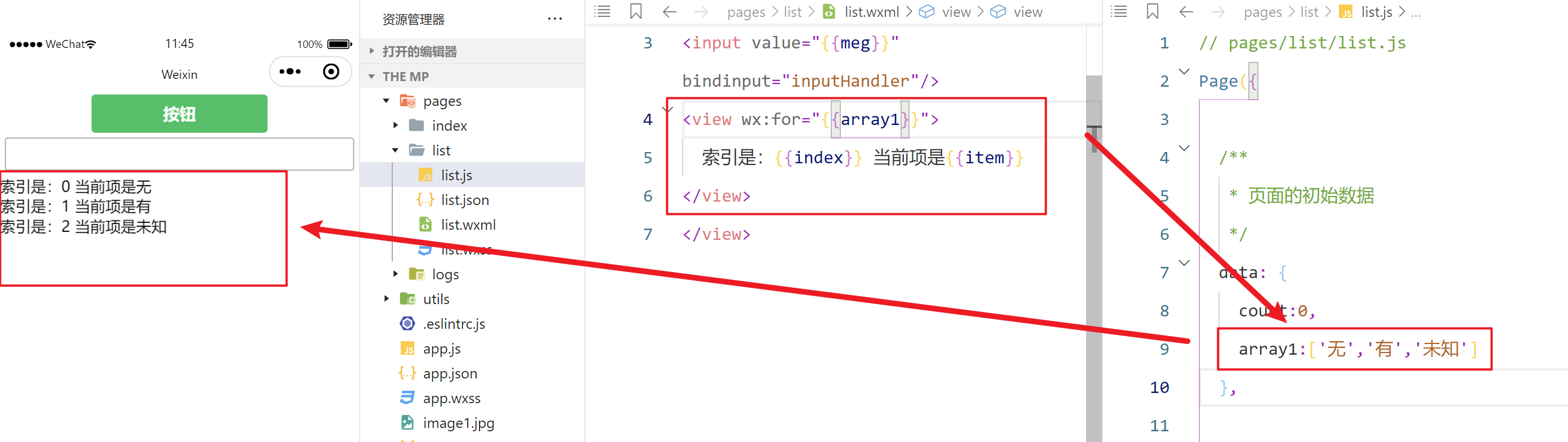
频繁切换建议使用后者

控制条件复杂时，建议使用wx:if搭配wx:elif、wx:else进行展示与隐藏的切换

## 6、列表渲染

### 6.1wx:for

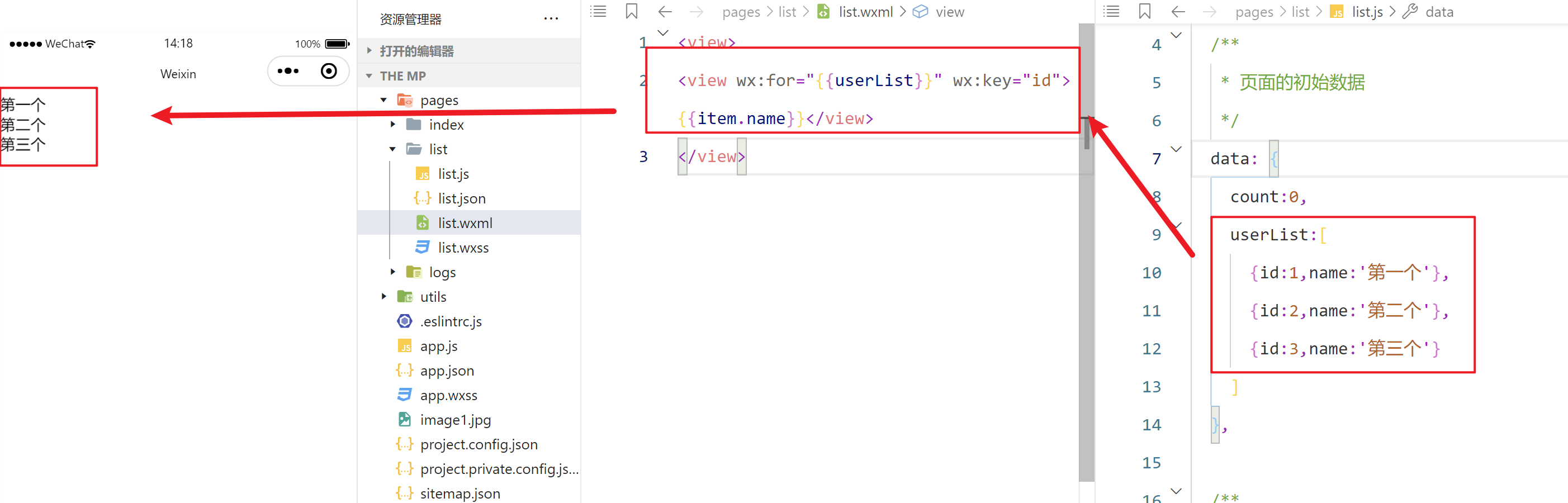
通过wx：for可以根据指定的数值，循环渲染重复的组件结构，如：



默认情况当前循环项的索引用index表示；当前循环项用item表示。

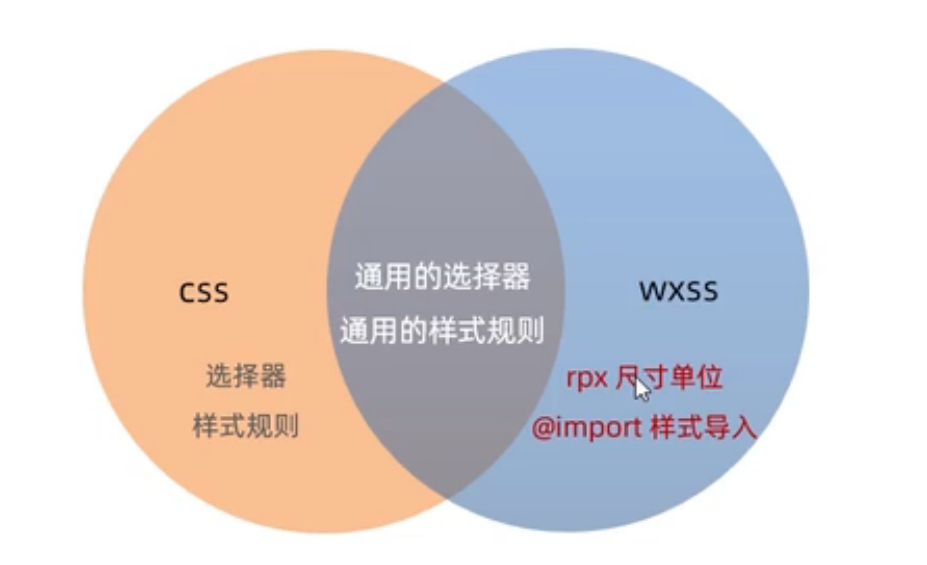
### 6.2wx:key的使用

类似于Vue列表渲染中的：key，小程序在实现列表渲染时，也建议为渲染出来的列表项指定唯一的key值，从而提高渲染效率



# 二、WXSS

wxss是一套样式语言，用于美化WXML的组件样式，类似于网页开发中的CSS。



## 1、模板样式

### 1.1rpx单位

rpx是微信小程序独有的，用来解决屏适配的尺寸单位。

### 1.2rpx的实现原理

鉴于不同设备屏幕的大小不同，为了实现屏幕的自动适配，px把所有设备的屏幕，在宽度上等分为750份(即：当前屏幕的总宽度为750rpx。

（小程序在不同设备上运行的时候，会自动把pX的样式单位换算成对应的像素单位来渲染，从而实现屏幕适配。

## 2、样式导入

使用WXSS提供的@import语法，可以导入外联的样式表。

@import后跟需要导入的外联样式表的相对路径，用；表示语句结束，如：

@import "common.WXSS";

### 2.1全局样式

定义在app.wxss中的样式为全局样式，作用于每一个页面

### 2.2 局部样式

在页面的.wxss文件中定义的样式为局部样式，只作用于当前页面。

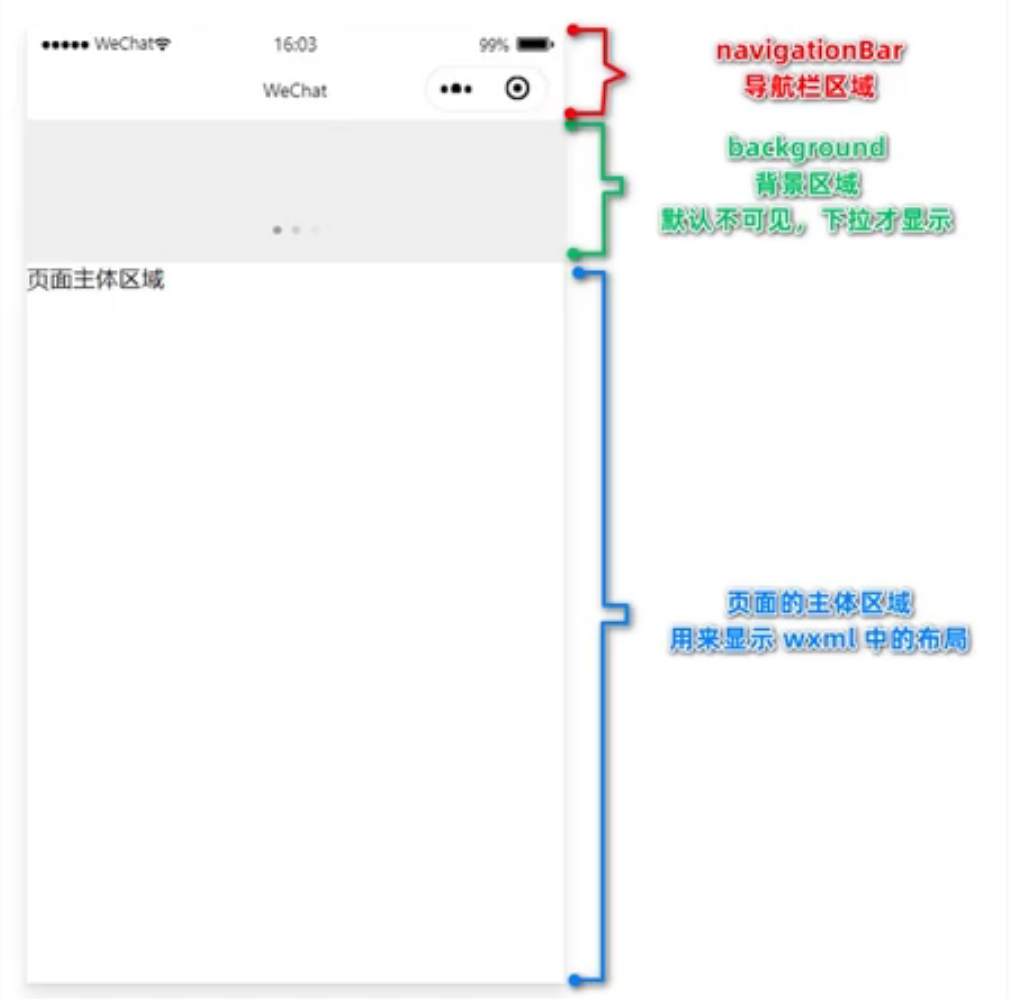
# 三、全局配置

## 1、全局配置文件及常用的配置项

小程序根目录下的app.json文件是小程序的全局配置文件。

常用配置如下：  
1）pages：记录当前小程序所有页面的存放路径  
2）window：全局设置小程序窗口的外观  
3）tabBar：设置小程序底部的tabBar效果  
4）style：是否启动新版的组件样式

## 2、小程序窗口组成部分

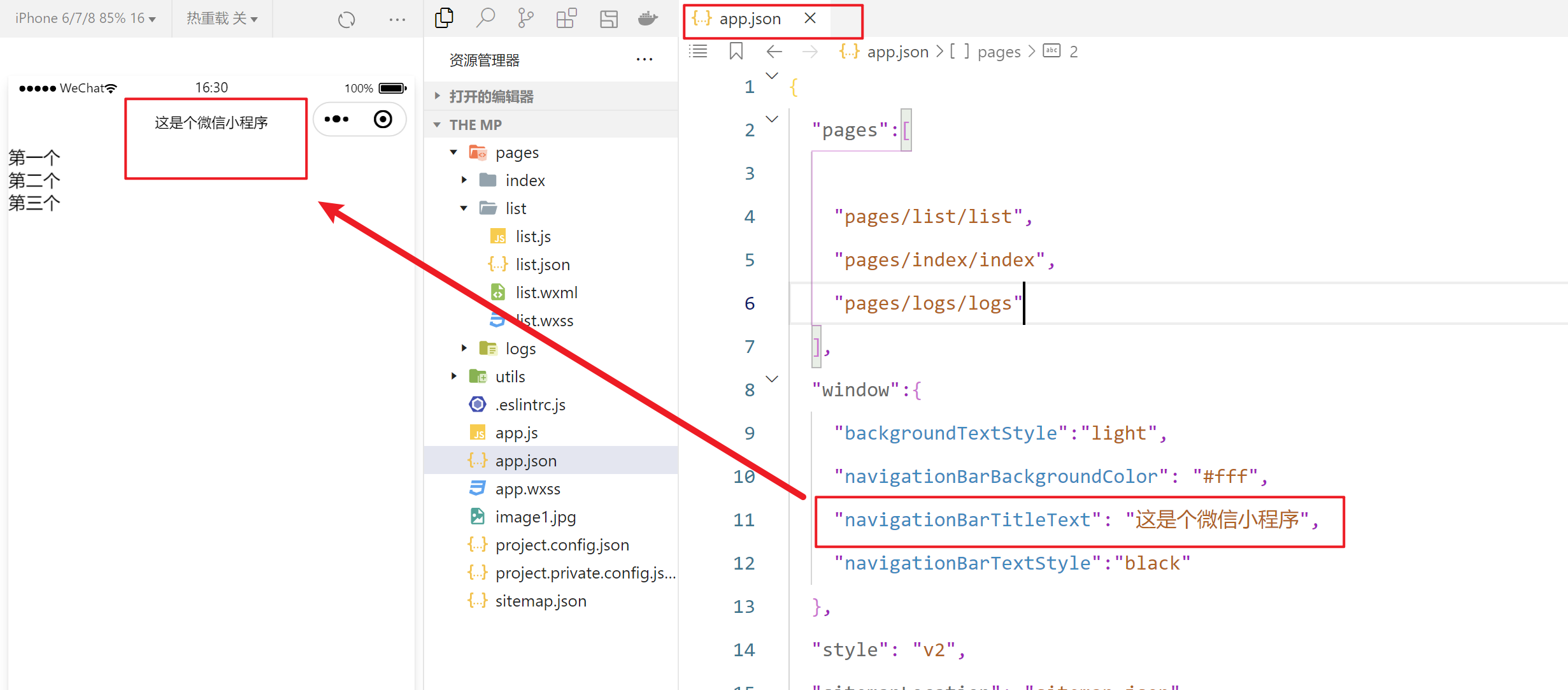


## 3、window节点常用的配置项

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 类型 | 默认值 | 说明 |
| navigationBarTitleText | String | 字符串 | 导航栏标题文字内容 |
| navigationBarBackgroundColor | HexColor | #000000 | 导航栏背景颜色，如#000000 |
| navigationBarTextStyle | String | White | 导航栏标题颜色，仅支持black/white |
| backgroundColor | HexColor | #ffffff | 窗口的背景色 |
| backgroundTextStyle | String | dark | 下拉loading的样式，仅支持dark/white |
| enablePullDowerRefresh | Bollean | flase | 是否全局开启下拉刷新 |
| onReachBottomDistance | Number | 50 | 页面上拉触底事件触发时距页面底部的距离，单位为px |

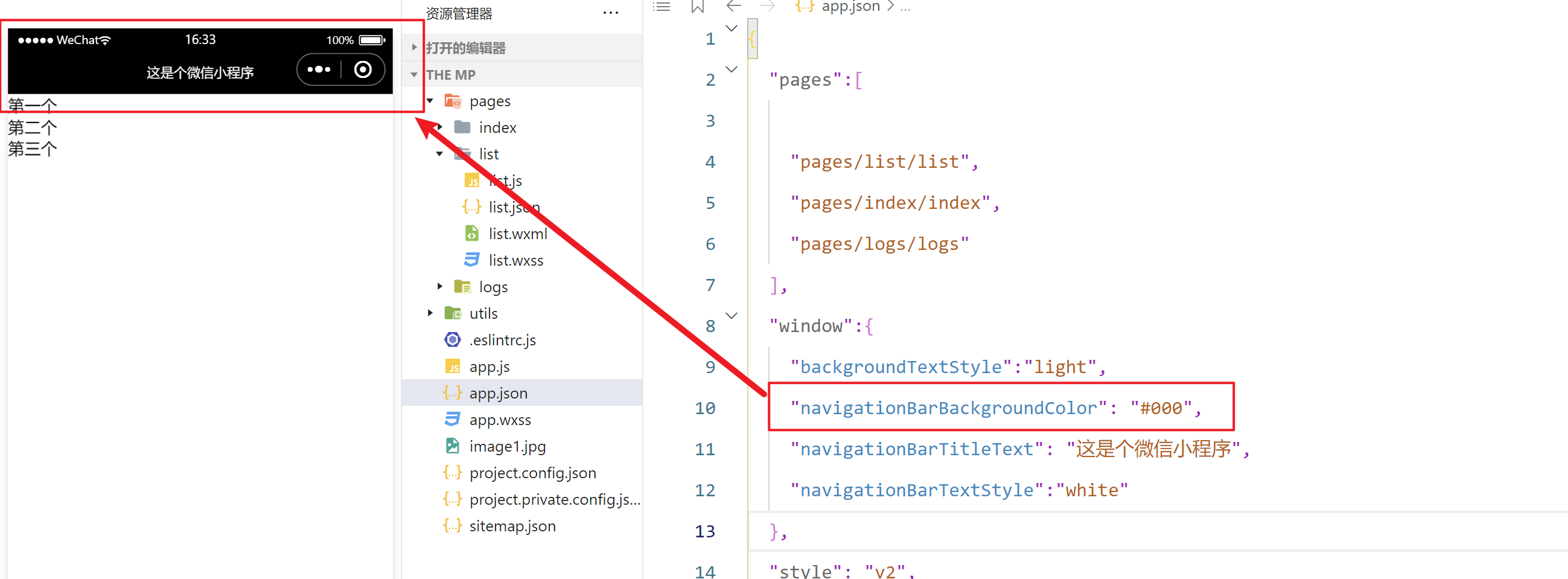
## 4、设置导航栏的标题

设置步骤：app.json->window->navigationBarTitleText



## 5、设置导航栏的背景色

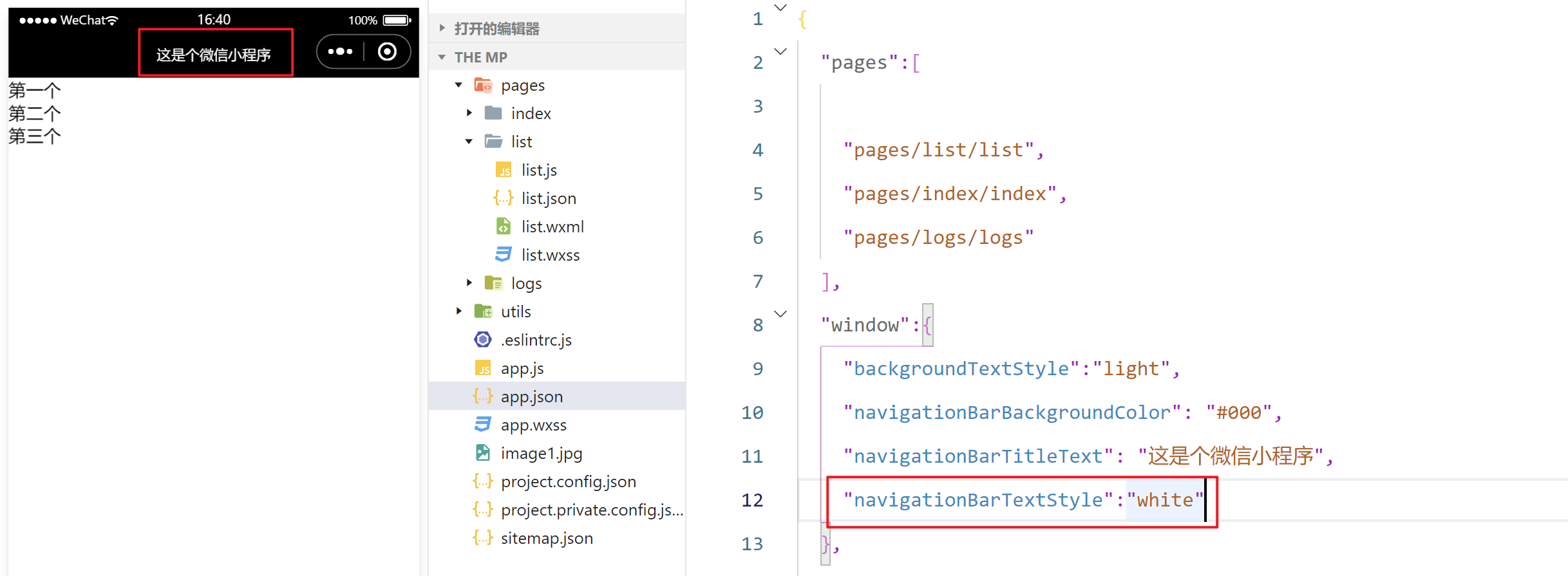
设置步骤：app.json->window->navigationBarBackgroundColor



## 6、设置导航栏的标题颜色

设置步骤：app.json->window->navigationBarTextStyle

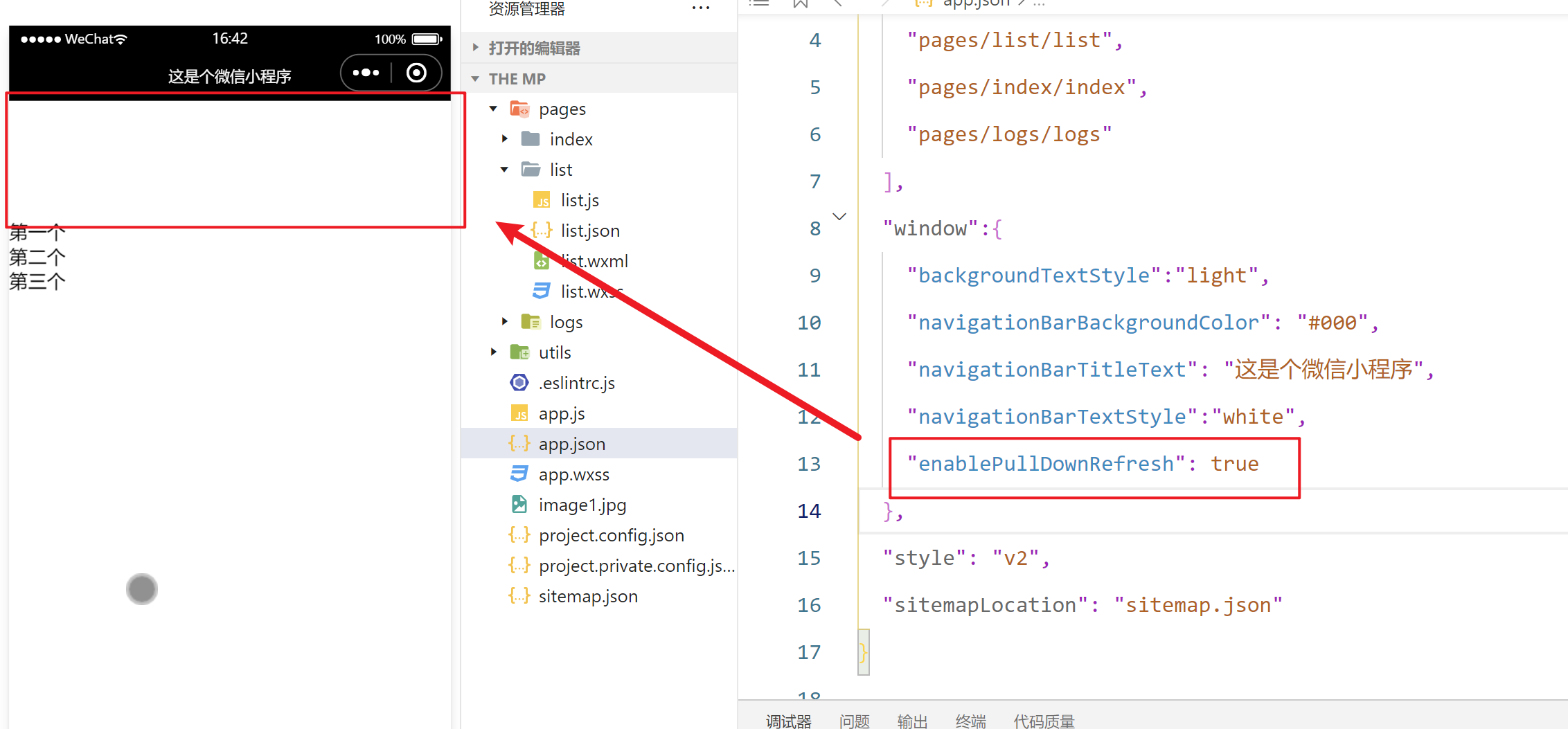
注意：navationBarTextStyle的可选值只有black和white



## 7、全局开启下拉刷新功能

概念：下拉刷新是移动端的专有名词，指的是通过手指在屏幕上的下拉滑动操作，从而重新加载页面数据的行为。

设置步骤：app.json->window->把enablePullDownRefresh的值设置为true

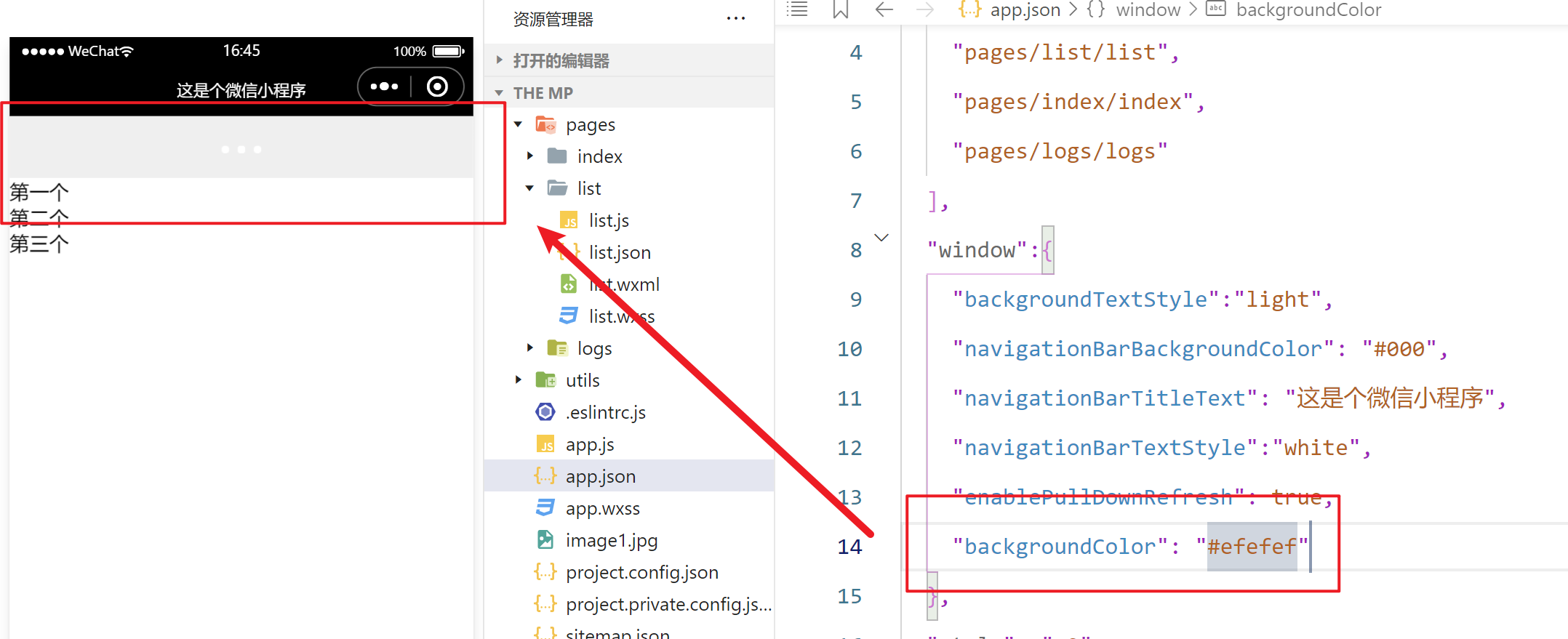


注意：在app.json中启用下拉刷新功能，会作用于每个小程序页面！

## 8、设置下拉刷新时窗口的背景色

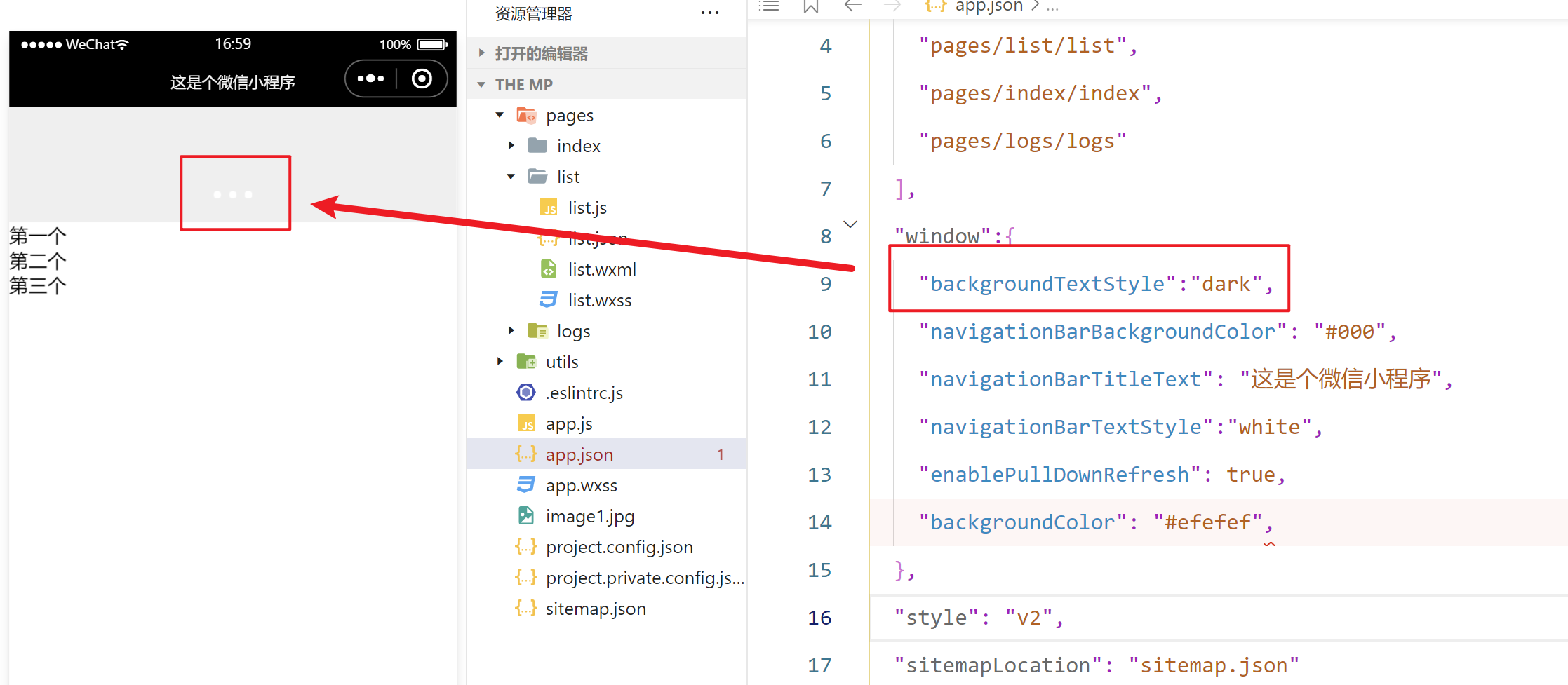
当全局开启下拉刷新功能之后，默认的窗口背景为白色。如果自定义下拉刷新窗口背景色，设置步骤为：

app.json->window->为backgroundColor指定16进制的颜色值如#efefef



## 9、设置下拉刷新时loading的样式

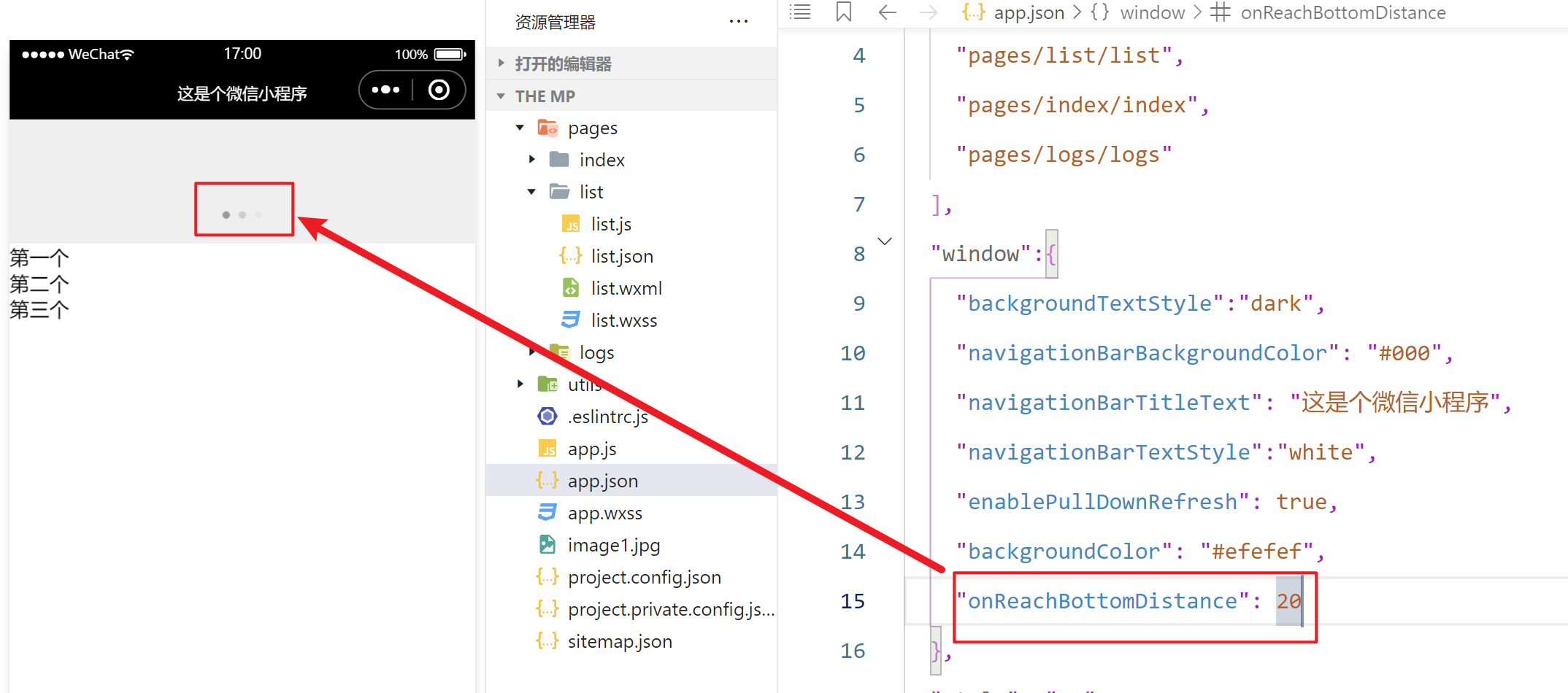
当全局开启下拉刷新功能之后，默认窗口的loading样式为白色，如果要更改loading样式的效果，设置步骤为app.json->window->为backgroundTextStyle指定dark值。



## 10、设置上拉触底的距离

概念：上拉触底是移动端的专有名词，通过手指在屏幕上的上拉滑动操作，从而加载更多数据的行为。

设置步骤：app.json->window->为onReachBottomDistance设置新的数值



注意：默认距离为50px，如果没有特殊需求，建议使用默认值即可。

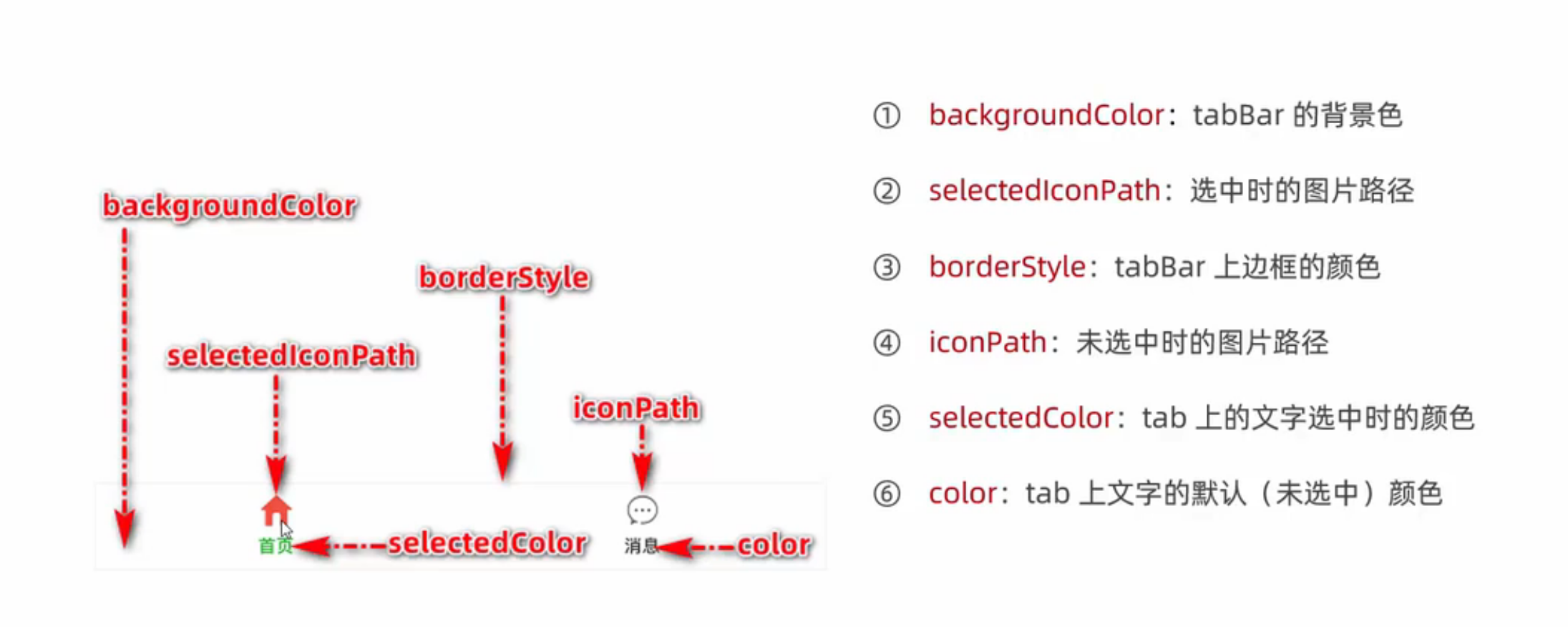
## 11、tabBar

tabBar是移动端应用常见的页面效果，用于实现多页面的快速切换。小程序中通常将其分为：底部tabBar和顶部tabBar

注意：

1）tabBar中只能配置最少2个，最多5个tab页签；

2）当渲染顶部tabBar时，不显示icon，只显示文本。



### 11.1 tabBar配置项

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性 | 类型 | 必填 | 默认值 | 描述 |
| position | String | 否 | bottom | TabBar的位置，仅支持borrom/top |
| borderStyle | String | 否 | black | tabBar上边框的颜色，仅支持black/white |
| color | HexColor | 否 |  | Tab上文字的默认（未选择）颜色 |
| selectedColor | HexColor | 否 |  | Tab 上文字选中时的颜色 |
| backgroundColor | HexColor | 否 |  | tabBar的背景色 |
| list | Array | 是 |  | Tab页签的列表，最少3个最多5个tab |

### 11.2每个tab项的配置选项

|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 类型 | 必填 | 描述 |
| pagePath | String | 是 | 页面路径，页面必须在pages中预先定义 |
| text | String | 是 | Tab上显示的文字 |
| iconPath | String | 否 | 未选中时的图标路径；当position为top时，不显示icon |
| selectediconParh | String | 否 | 选中时的图标路径；当position为top时，不显示icon |

注意：阿里图标库可以下载icon图标，tabBar的相关页面必须放到“pages”的最前面的位置！不然无法显示。

# 四、页面配置

小程序中，每个页面都要自己的.json配置文件，用来对当前页面的窗口外观、页面效果等进行配置。

页面配置中常见配置项：

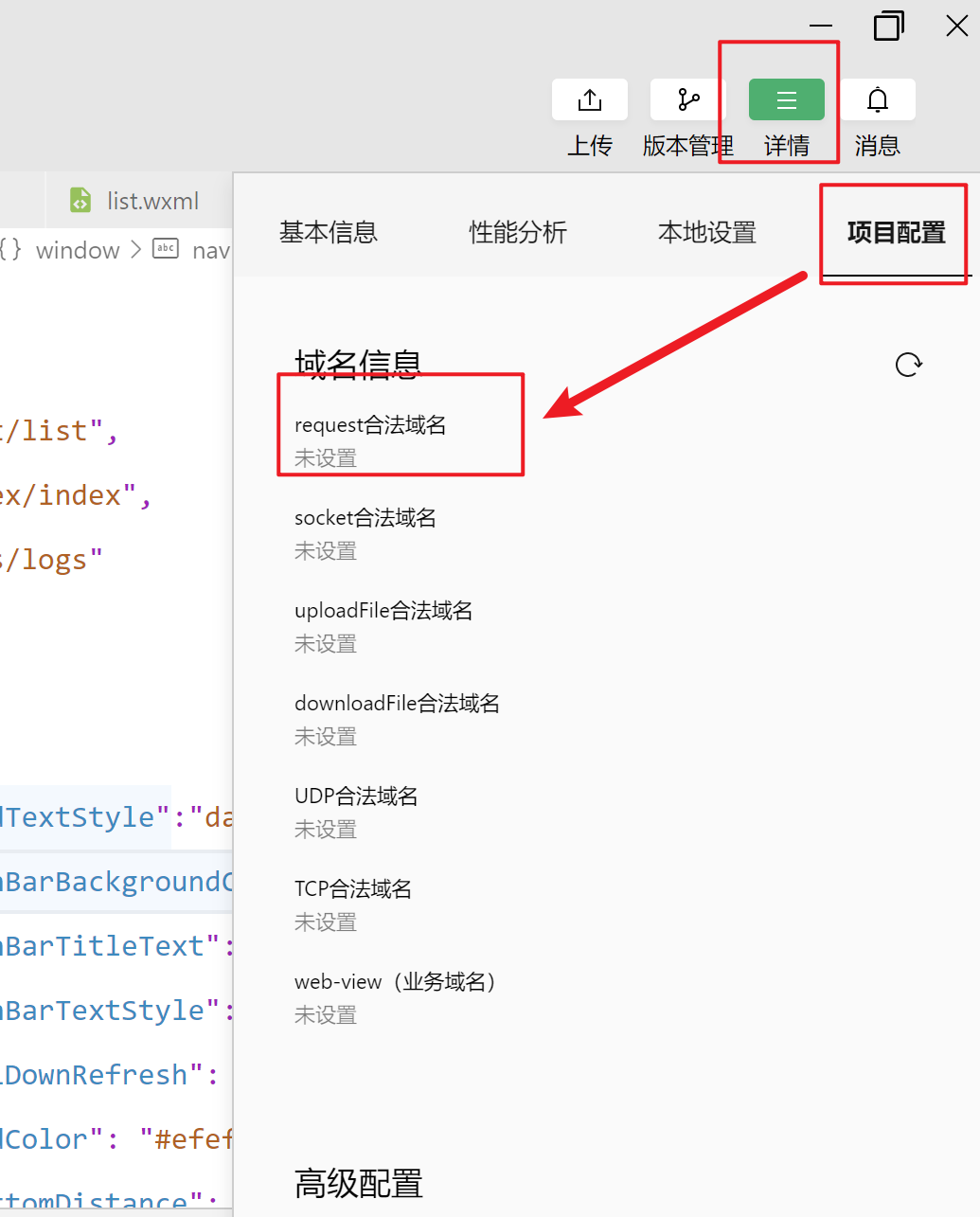
|  |  |  |  |
| --- | --- | --- | --- |
| 属性 | 类型 | 默认值 | 描述 |
| navugationBargroundColor | Hexcolor | #000000 | 当前页面导航栏背景颜色 |
| navigatioBarTextStyle | Sting | white | 当前页面导航栏标题颜色，仅支持black/white |
| navigationBarTitileText | String |  | 当前页面导航栏标题文字内容 |
| backgroundColor | HexColor | #ffffff | 当前页面窗口的背景色 |
| backgroundTextStyle | String | dark | 当前页面下拉loading的样式。仅支持dark/light |
| enablePullDownRefresh | Boolean | fasle | 是否为当前页面开启下来刷新的效果 |
| onReachBottomDistance | Number | 50 | 页面上拉触底事件触发时距页面底部距离，单位为px |

# 五、网络数据请求

小程序官方对数据接口的请求做了两个限制：

1）只能请求HTTPS类型的接口；

2）必须将接口的域名添加到信任列表中。



## 1、配置request合法域名

配置步骤：登录微信小程序管理后台->开发->开发设置->服务器域名->修改request合法域名

注意事项：  
1）域名只支持https协议；  
2）域名不能使用IP地址或localhost；  
3）域名必须经过ICP备案；  
4）服务器域名一个月最多可申请修改5次；

## 2、发起GET和POST请求

调用微信小程序提供的wx.request（）方法，可以发起GET数据请求.

调用微信小程序提供的wx.request（）方法，可以发起POST数据请求.

## 3、在页面刚加载时请求数据

需要在页面刚加载的时候，自动请求一些初始化的数据。此时需要在页面的onLoad事件中调用获取数据的函数。如：

