

---

## 第 3 章 我的页面模块开发

本章主要通过我的页面模块开发正式开始豆豆云助教的开发。完成我的页面开发之前，需要先完成授权登录页面和注册页面，拥有授权信息与注册信息后，才能在我的页面将个人信息显示出来。

### 3.1 授权登录页面

本节主要分为两个部分，首先是讲解一下授权登录页面涉及到的知识点，然后在理解的情况下完成授权登录页面的开发。

#### 3.1.1 授权页面知识点讲解

##### 1. 小程序登录

小程序可以通过微信官方提供的登录能力方便地获取微信提供的用户身份标识，快速建立小程序内的用户体系。如图 3-1 所示，小程序通过 `wx.login()` 获取 `code`，然后通过 `wx.request()` 发送 `code` 至开发者服务器，开发者服务器将登录凭证 `appid`、`appsecret` 与 `code` 用于校验微信接口，微信接口服务向开发者服务器返回用户唯一标识 `OpenID` 和 会话密钥 `session_key`。开发者服务器实现自定义登录状态与 `openid` 与 `session_key` 关联，并向小程序返回自定义状态。小程序将自定义登录状态存入 `storage`，并用于后续 `wx.request` 发起业务请求。

对于某个微信小程序，每个用户访问该小程序都有产生一个唯一的 `openid`，这个 `openid` 为用户访问该小程序的标识符，即每个用户的 `openid` 都是不一样的。因此，可以把 `openid` 作为用户唯一标识符（类似身份证号），并存于数据库中用以后续操作。

开发者服务器与微信接口服务之间的交互是由后台实现的，本节主要以小程序前端与开发者服务器之间的交互为主，后台部分会在第 9 章中进行详细介绍。

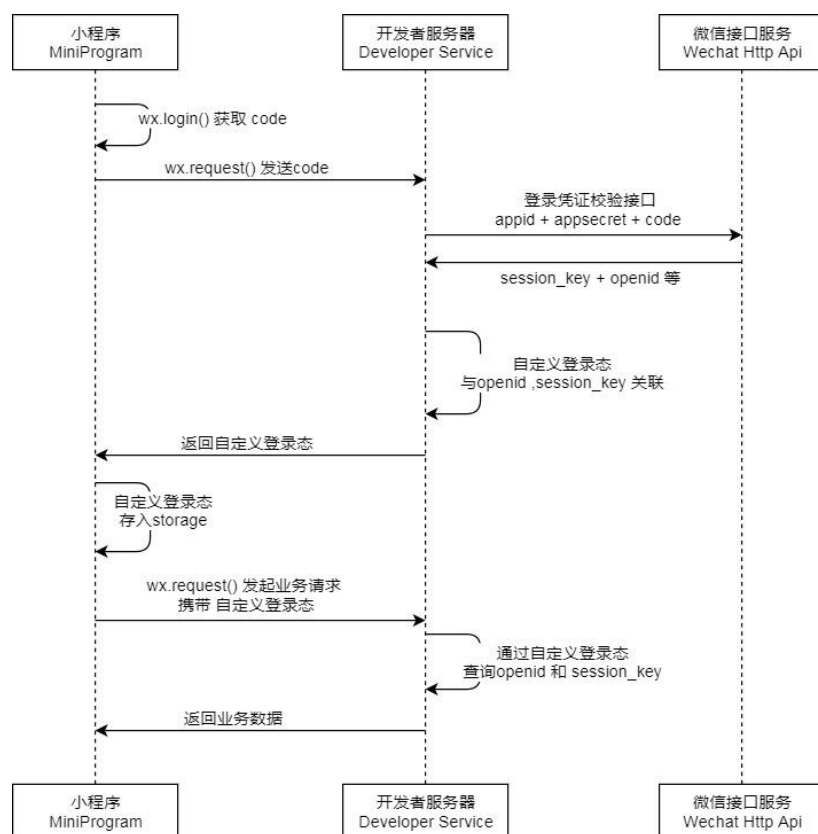


图 3 - 1 小程序登录流程时序

## 2. wx.login( )

调用 `wx.login( )` 接口获取登录凭证（code），通过凭证进而换取用户登录态信息，其中 `wx.login( )` 接口的属性如表 3 - 1 所示。

表 3 - 1 wx.login()接口属性

属性	类型	必填	说明
timeout	number	否	超时时间，单位 ms
success	function	否	接口调用成功的回调函数
fail	function	否	接口调用失败的回调函数
complete	function	否	接口调用结束的回调函数（调用成功失败都会执行）

由于 `app.js` 会先于其他页面执行，所以比较适合处理一些注册函数，因此将 `wx.login( )` 方法写在 `app.js` 文件中。

## 3. wx.request( )

`wx.request( )` 主要用于发送 https 网络请求，其属性详见表 3 - 2。

表 3 - 2 wx.request 属性表

属性	类型	默认值	必填	说明
url	string		是	开发者服务器接口地址
data	string/object/ ArrayBuffer		否	请求参数
header	object		否	设置请求的 header，header 中不能设置 Referer。content-type 默认为 application/json
method	string	GET	否	HTTP 请求方法
dataType	string	json	否	返回的数据格式
responseType	string	text	否	响应的数据类型
success	function		否	接口调用成功的回调函数
fail	function		否	接口调用失败的回调函数
complete	function		否	接口调用结束的回调函数（调用成功失败都会执行）

这里以小程序登录中，小程序向开发者服务器发送 wx.request 请求为例，调用微信官方的 wx.login()接口会返回一串 jscode，服务器使用 jscode、appid、appsecret 三个参数向微信请求得到 openid，这一步后台已经封装完成，并提供一个开放接口：

[https://zjgsujiaoxue.applinzi.com/index.php/Api/Weixin/code\\_to\\_openidv2](https://zjgsujiaoxue.applinzi.com/index.php/Api/Weixin/code_to_openidv2)

具体代码如下：

```
// 登录
wx.login({
  success: res => {
    // 发送 res.code 到后台换取 openId, sessionKey, unionId
    wx.request({
      url:
'https://zjgsujiaoxue.applinzi.com/index.php/Api/Weixin/code_to_openidv2',
      data: {
        'code': res.code,
        'from': 'wxbf9778a9934310a1'
      },
      success: function (res) {
        console.log(res.data)
        //将 SESSIONID 保存到本地 storage
```

```

        wx.setStorageSync('jiaoxue_OPENID', res.data.openid)
    },
    fail: function (res) {
        console.log('res' + res)
    }
})
}
})

```

上述代码中，通过 `wx.login()` 方法，成功返回 `res`，其中 `res.code` 为微信官方返回 `code`，通过 `wx.request` 请求，请求参数为 `code` 与 `appid`，向请求成功时，后台会返回一个数组，数组中包含的值是有后台代码决定的，其中就包含了 `openid`，这里可以使用 `console.log(res.data)` 来看一下返回的数组中所包含的值，如图 3 - 2 所示。



图 3 - 2 wx.request 请求的返回值

#### 4. 数据缓存

每个微信小程序都可以有自己的本地缓存，可以通过数据缓存 API 可以对本地缓存进行设置、获取和清理。同一个微信用户，同一个小程序 `storage` 上限为 10MB。`localStorage` 以用户维度隔离，同一台设备上，A 用户无法读取到 B 用户的数据。

注意：如果用户储存空间不足，微信会清空最近最久未使用的小程序的本地缓存。因此不建议将关键信息全部存在 `localStorage`，以防储存空间不足或用户换设备的情况。

数据缓存 API 主要有 5 类，包括数据的存储、获取、移除、清空，以及获取存储信息，每类均包含同步与异步两种，具体详见表 3 - 3。

表 3 - 3 数据缓存 API 函数类型

函数名	说明
<code>wx.setStorage(Object object)</code>	数据的存储（异步）
<code>wx.setStorageSync(string key, any data)</code>	数据的存储（同步）

wx.getStorage(Object object)	数据的获取（异步）
wx.getStorageSync(string key)	数据的获取（同步）
wx.getStorageInfo(Object object)	存储信息的获取（异步）
wx.getStorageInfoSync()	存储信息的获取（同步）
wx.removeStorage(Object object)	数据的移除（异步）
wx.removeStorageSync(string key)	数据的移除（同步）
wx.clearStorage(Object object)	数据的清空（异步）
wx.clearStorageSync()	数据的清空（同步）

其中 Sync 为英文单词 synchronization 的前四个字母，表示同步，因此 API 函数中带有 Sync 后缀的函数为同步函数。同步函数与异步函数之间的区别是，异步不会阻塞当前任务，同步缓存直到同步方法处理完才能继续往下执行。另外异步函数中含有成功回调函数，可用于表示数据处理成功后的操作。

这里以 wx.login() 中使用的 wx.setStorage() 为例，将通过 wx.request() 返回的 openid 存储于本地，方便 openid 的获取，使用 wx.setStorage() 的代码示例如下：

```
wx.setStorageSync('jiaoxue_OPENID', res.data.openid)
```

编译后，可以在调试器的 storage 面板中看到 openid 以存入本地，Key 的值为 jiaoxue\_OPENID, Value 的值为用户的 openid，如图 3 - 3 所示。

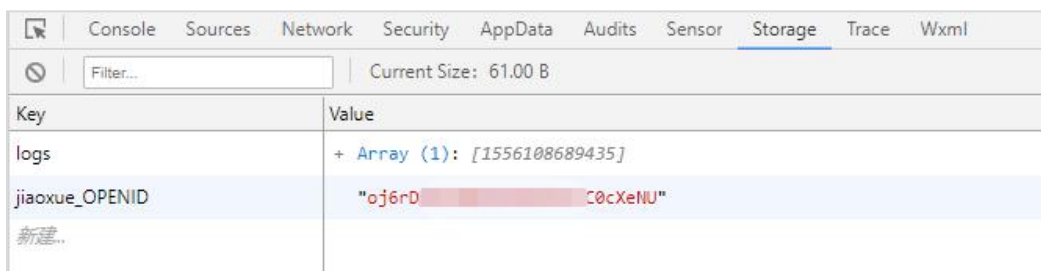


图 3 - 3 storage 面板中的本地缓存

如果使用 wx.setStorageSync() 进行数据存储，可以对数据存储成功后进行操作，代码较 wx.setStorage() 有变化，具体代码如下：

```
wx.setStorage({
  key: 'test',
  data: 'begin',
  success: function(){
    console.log('存储成功')
  }
})
```

编译后，同样将 `openid` 存储于本地缓存，并执行成功回调函数，打印出“存储成功”表示存储成功，如图 3 - 4 所示。



图 3 - 4 console 面板中的“存储成功”

需要使用本地缓存中的 `openid` 时，可以用 `wx.getStorageSync('jiaoxue_OPENID')` 来获取 `openid`，并赋值给相应的变量。当然 `wx.getStorage` 也可以，这里不过多赘述。

### 5. `wx.showModal()`

小程序使用 `wx.showModal(Object object)` 显示模态对话框，其中 `object` 参数说明如表 3 - 4 所示。

表 3 - 4 `wx.showModal` 的参数

属性	类型	默认值	必填	说明
<code>title</code>	<code>string</code>		是	提示的标题
<code>content</code>	<code>string</code>		是	提示的内容
<code>showCancel</code>	<code>boolean</code>	<code>true</code>	否	是否显示取消按钮
<code>cancelText</code>	<code>string</code>	‘取消’	否	取消按钮的文字，最多 4 个字符
<code>cancelColor</code>	<code>string</code>	<code>#000000</code>	否	取消按钮的文字颜色，必须是 16 进制格式的颜色字符串
<code>confirmText</code>	<code>string</code>	‘确定’	否	确认按钮的文字，最多 4 个字符
<code>confirmColor</code>	<code>string</code>	<code>#576B95</code>	否	确认按钮的文字颜色，必须是 16 进制格式的颜色字符串
<code>success</code>	<code>function</code>		否	接口调用成功的回调函数
<code>fail</code>	<code>function</code>		否	接口调用失败的回调函数
<code>complete</code>	<code>function</code>		否	接口调用结束的回调函数（调用成功、失败都会执行）

其中 `success` 回调函数的返回参数，详见表 3 - 5。

表 3 - 5 success()返回参数

属性	类型	说明	最低版本
confirm	boolean	为 true 时，表示用户点击了确定按钮	
cancel	boolean	为 true 时，表示用户点击了取消（用于 Android 系统区分点击蒙层关闭还是点击取消按钮关闭）	1.0.0

在进入豆豆云助教时，如果用户没有注册过，会弹出模态弹窗提示用户前往注册，具体代码如下：

```
if (!res.data.is_register) {  
  wx.showModal({  
    title: '提示',  
    content: '请先注册',  
    showCancel: false,  
    confirmText: "确定",  
    success: function(res) {  
      wx.navigateTo({  
        url: '/pages/register/userlogin',  
      })  
    }  
  })  
}
```

编译后，弹出模态对话框，提示用户前往注册，如图 3 - 5 所示。

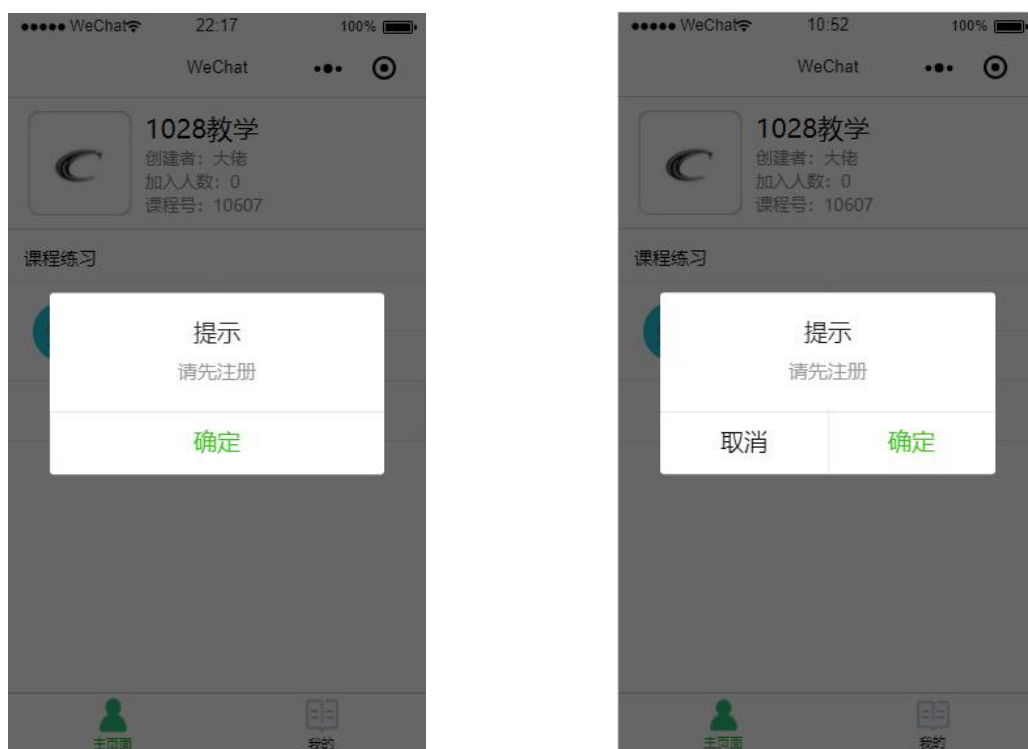


图 3 - 5 模态对话框提示注册（不含取消按钮）

图 3 - 6 模态对话框提示注册（含取消按钮）

另外尝试在该 `wx.showModal()` 的基础上，进行简单的修改，首先将 `showCancel` 属性删除，这样模态弹窗会默认 `showCancel` 的值为“true”。然后添加一个成功回调函数 `success()`，通过 `console.log()` 查看一下 `success()` 的返回值具体有哪些，具体代码如下。

```
wx.showModal({
  title: '提示',
  content: '请先注册',
  confirmText: "确定",
  success: function(res) {
    console.log(res)
    if(res.confirm){
      console.log('“确定”按钮被单击')
      wx.navigateTo({
        url: '/pages/register/userlogin',
      })
    }else if(res.cancel){
      console.log('“取消”按钮被单击')
    }
  }
})
```

编译后，效果图如图 3 - 6 所示。在 Console 面板中可以看到打印出来的 `success` 函数的返回值，如图 3 - 7 所示。

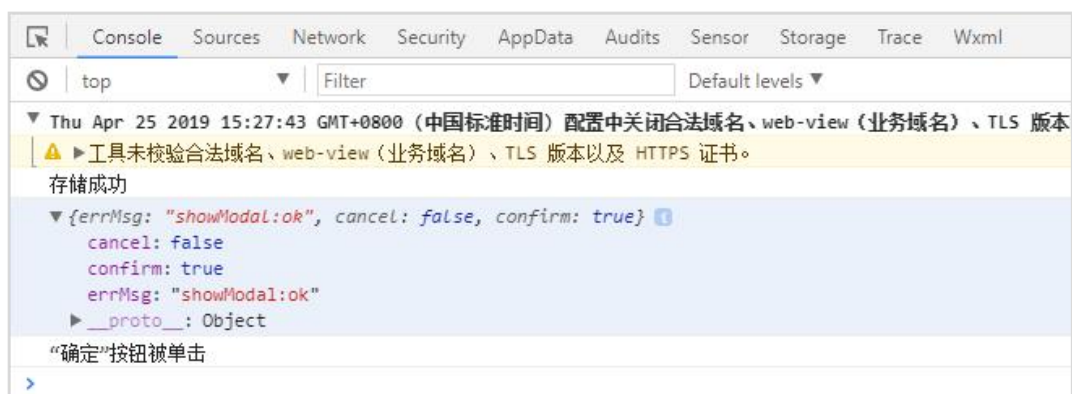


图 3 - 7 success()返回值

### 3.1.2 授权登录页面实现

#### 1. 新建小程序项目

首先新建一个小程序项目，具体操作与 1.1.3 中 hello world 小程序的新建一样，



新建项目时，建议开发者自定义项目名称，并且自己在存放小程序项目的目录下新建一个空的文件夹，选择项目目录时，选择该文件夹，这样方便之后寻找项目。比如项目名称命名为“doudouyun”，与项目相关，具体如图 3 - 8 所示。

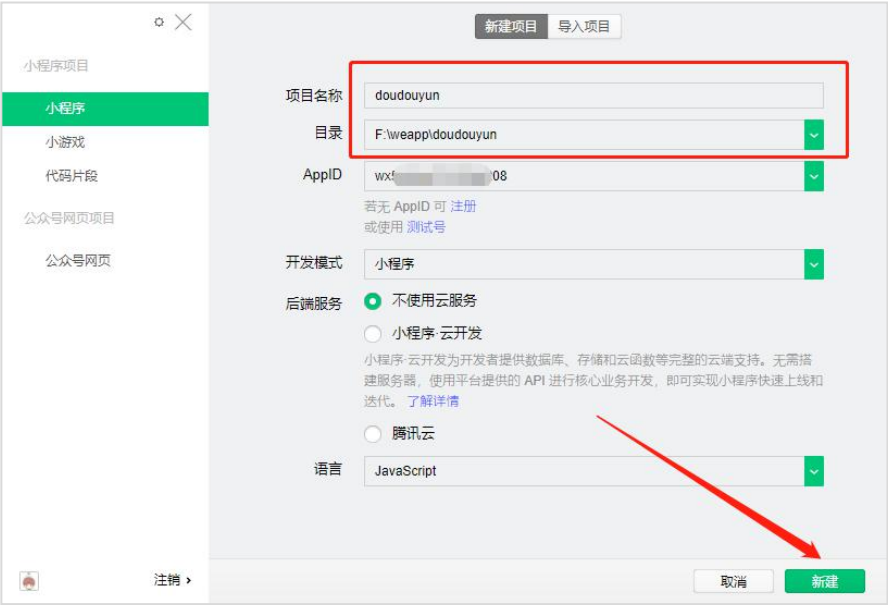


图 3 - 8 新建豆豆云项目

## 2. 新建 userlogin 页面

完成项目新建后，需要新建一个授权登录页面，首先右击“pages”，单击“新建目录”按钮，并命名为“register”。然后右击“register”目录，单击“新建 Page”按钮，并命名为“userlogin”，如图 3 - 9 和图 3 - 10 所示。

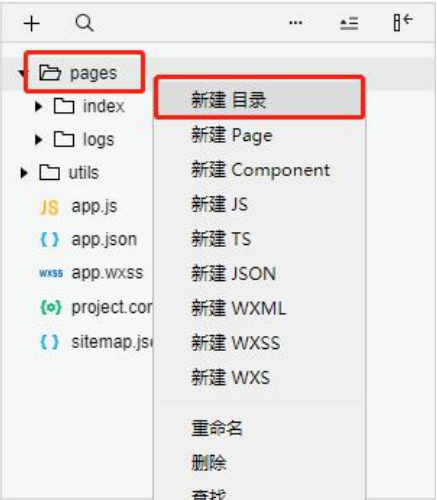


图 3 - 9 新建 register 目录

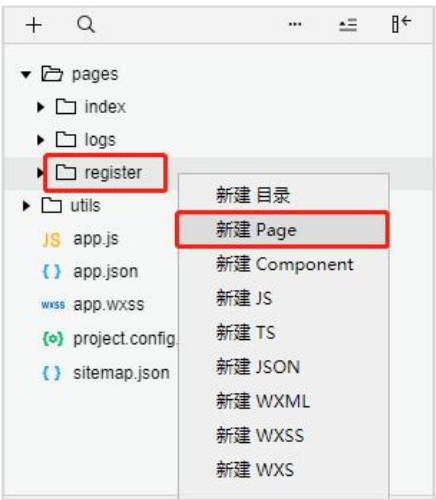


图 3 - 10 新建 userlogin 页面

选择“新建 Page”而不选择一个一个文件新建，原因是选择“新建 Page”时，app.json 的 pages 属性中会自动添加新建的页面，开发者不需要再手动添加页面路

---

径了。

### 3. userlogin 页面开发

userlogin 页面的功能主要是授权的功能，与 hello world 小程序中 index 页面的功能相似，因此只要在 hello world 小程序的基础上，进行简单修改就行。

首先是 wxml 文件，userlogin 页面结构主要由 view、text 与 button 三种标签组成，并使用 class 属性定义对应标签的样式，页面主要是一个“点击授权登录”按钮，具体代码如下。

```
<!--userlogin.wxml-->
<view class="container">
  <view class="usermotto">
    <text class="user-motto">微信授权</text>
  </view>
  <view class="userinfo">
    <button wx:if="{{!hasUserInfo && canIUse}}" open-type="getUserInfo"
      bindgetuserinfo="getUserInfo">点击授权登录</button>
  </view>
</view>
```

然后是 wxss 文件，相比于 hello world 小程序中 index.wxss 文件，少了两种样式类型，主要保留了 userinfo 与 usermotto，具体代码如下。

```
/**userlogin.wxss*/
.userinfo {
  display: flex;
  flex-direction: column;
  align-items: center;
}
.usermotto {
  margin-top: 150px;
  text-align: center;
}
```

为了用户更好的体验，一些小细节也要注意一下，比如当用户进入授权登录页面时，页面导航栏的标题文字也相应变为“授权页面”，主要就是在 json 文件中加上一行代码，具体代码如下。

```
{
  "navigationBarTitleText": "授权页面"
}
```

最后就是 userlogin.js 中的相关逻辑代码，userlogin 页面的逻辑与 hello world 小程序中 index 页面的逻辑基本一样，只是简单调整了一下，原有的事件处理函数

`bindViewTap` 在授权页面不需要了，直接删了就行。然后就是在 `onLoad` 函数最后加上一个判断语句，判断当 `hasUserInfo != false` 时，跳转至 `register` 页面，即注册页面，具体代码如下。

```
if (this.data.hasUserInfo) {  
  wx.navigateTo({  
    url: './register',  
  })  
}
```

另外 `getUserInfo` 函数中也相应加上一个页面跳转 `wx.navigateTo()`，实现当触发事件处理函数 `getUserInfo` 时，跳转至 `register` 页面，具体代码如下。

```
getUserInfo: function (e) {  
  wx.navigateTo({  
    url: './register',  
  })  
  app.globalData.userInfo = e.detail.userInfo  
  this.setData({  
    userInfo: e.detail.userInfo,  
    hasUserInfo: true  
  })  
}
```

最后授权登录页面的效果图如图 3 - 11 所示。



图 3 - 11 授权登录页面效果图

如果发现之前已经授权过了，看不到想要的授权页面，可以单击工具栏中间区域的“清缓存”按钮，来清除授权记录。

#### 4. app.js

除了完成 userlogin 页面的开发,还需要对 app.js 文件进行修改,首先是 wx.login() 方法中需要完善,才能实现小程序登录功能,最终代码如下。

```
wx.login({
  success: res => {
    // 发送 res.code 到后台换取 openid, sessionKey, unionId
    wx.request({
      url: 'https://zjgsujiaoxue.applinzi.com/index.php/Api/Weixin/code_to_openidv2',
      data: {
        'code': res.code,
        'from': 'wx5ee2da791099a208'
      },
    },
    success: function (res) {
      console.log(res.data)
      //将 SESSIONID 保存到本地 storage
      wx.setStorageSync('jiaoxue_OPENID', res.data.openid)
      if (!res.data.is_register) {
        wx.showModal({
          title: '提示',
          content: '请先注册',
          showCancel: false,
          confirmText: "确定",
          success: function (res) {
            wx.navigateTo({
              url: '/pages/register/userlogin',
            })
          }
        })
      }
    },
    fail: function (res) {
      console.log('res' + res)
    }
  })
})
```

注意, wx.request() 的 data 数组中, from 对应的是 appid, 因此后面 appid 的值需要改成开发者自己的 appid。

编译一下后, 会发现 Console 面板会提示错误, 如图 3 - 12 所示。

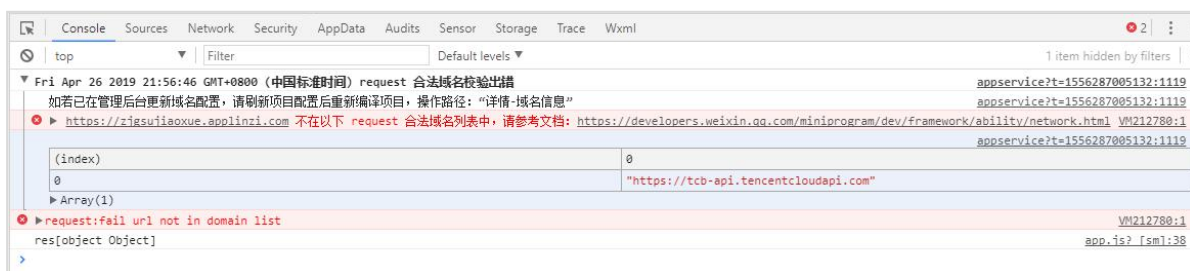


图 3 - 12 提示 request 中 url 不在合法域名列表

解决方法：单击工具栏右侧区域的“详情”按钮，勾选“不校验合法域名”即可，如图 3 - 13 所示。

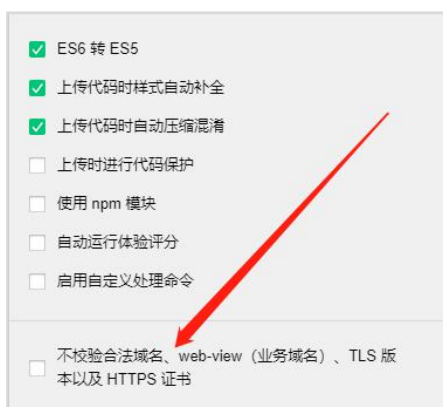


图 3 - 13 勾选“不校验合法域名”

勾选“不校验合法域名”后，重新编译一次，发现 Console 面板提示“该 appid 未注册”，如图 3 - 14 所示。



图 3 - 14 提示“该 appid 未注册”

这是因为为了让所有开发者在学习豆豆云前端开发时，使用提供给所有开发者的云后台，豆豆云开发者为开发者专门写了一个接口，前往注册一下，即可使用提供的云后台。因此要使 `wx.login` 方法的 `wx.request()` 中的 url 实现访问后台，需要前往 <https://zjgsujiaoxue.applinzi.com/index.php/Page/Index/register> 进行使用注册。调用该接口需要 2 个参数，即开发者的 appid 与 appsecret，如图 3 - 15 所示。

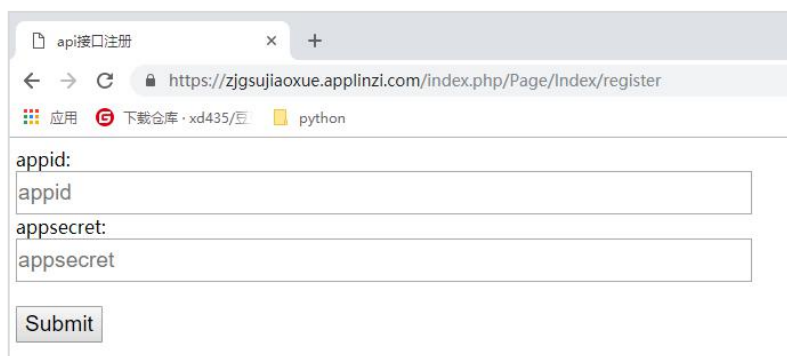


图 3 - 15 api 接口注册

填写 appid 与 appsecret 后，单击“Submit”按钮即可完成 api 接口注册。api 接口注册完成后，重新编译代码即可看到 Console 面板中 wx.request() 的返回值，主要包括了 is\_login、is\_register 和 openid，如图 3 - 16 所示。



图 3 - 16 wx.request() 返回值

到这里，用户的第一次进入到授权登录页面的跳转逻辑已经完成。

## 3.2 注册页面

在 userlogin 的逻辑中要跳转到注册页面，那就需要新建一个 register 页面。本节主要先对注册页面中的一些知识点进行讲解，然后再具体介绍如何完成注册页面的开发。

### 3.2.1 注册页面知识点讲解

注册页面主要新增了三个知识点，分别是微信官方 UI 库 WeUI，bindchange 事件和 openAlert 函数。

#### 1. 微信官方 UI 库 WeUI

WeUI 是一套同微信原生视觉体验一致的基础样式库，由微信官方设计团队为微信内网页和微信小程序量身设计，令用户的使用感知更加统一。包含 button、cell、dialog、progress、toast、article、actionsheet、icon 等各式元素。WeUI 基础样式库下载地址：<https://github.com/Tencent/weui-wxss>。开发者可以将样式库下载并使用微信 web 开发者工具打开 dist 目录（请注意，是 dist 目录，不是整个项目），导入 dist 目录后，可以预览样式库，如图 3 - 17 所示。



图 3 - 17 WeUI 样式库预览

开发者可以在样式库里选择自己所需要的样式，然后将需要的样式的 wxml 复制黏贴至自己的项目中，然后将 WeUI 中 style 文件拷贝至自己的项目目录中，如将图 3 - 18 目录下 style 文件夹拷贝至图 3 - 19 目录下。

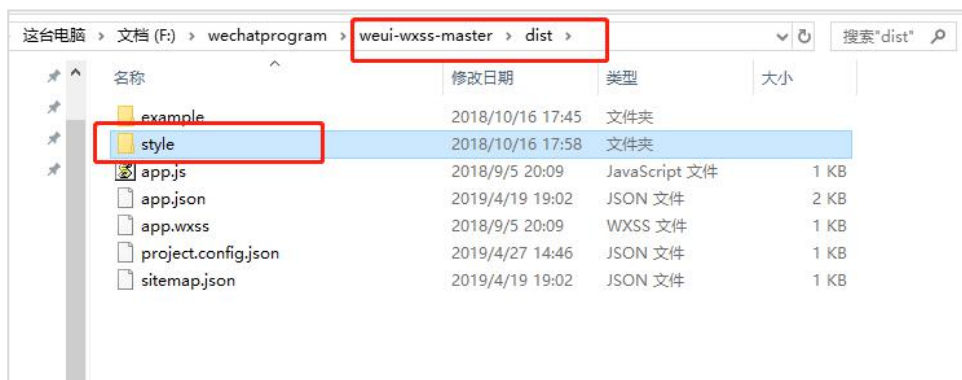


图 3 - 18 dist 目录下的 style 文件夹



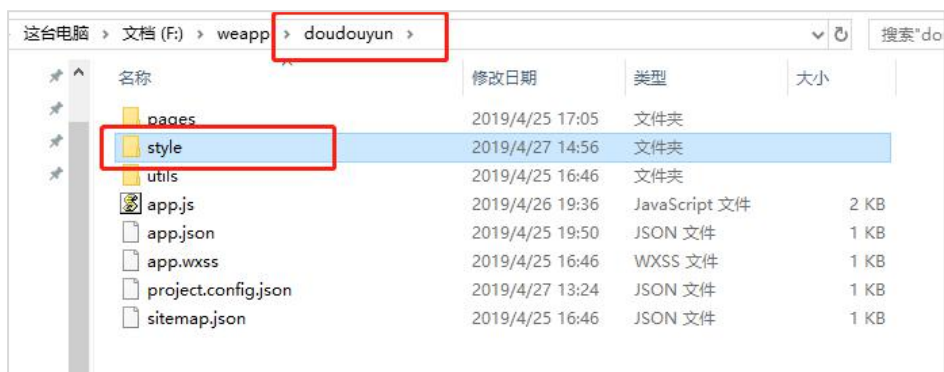


图 3 - 19 doudouyun 项目下的 style 文件夹

将 style 文件夹拷贝至自己开发的项目后，还需要在 app.wxss 文件中使用 @import 导入 weui 的样式，如图 3 - 20 所示。到这里，就可以正常使用 WeUI 库中微信的官方样式了。

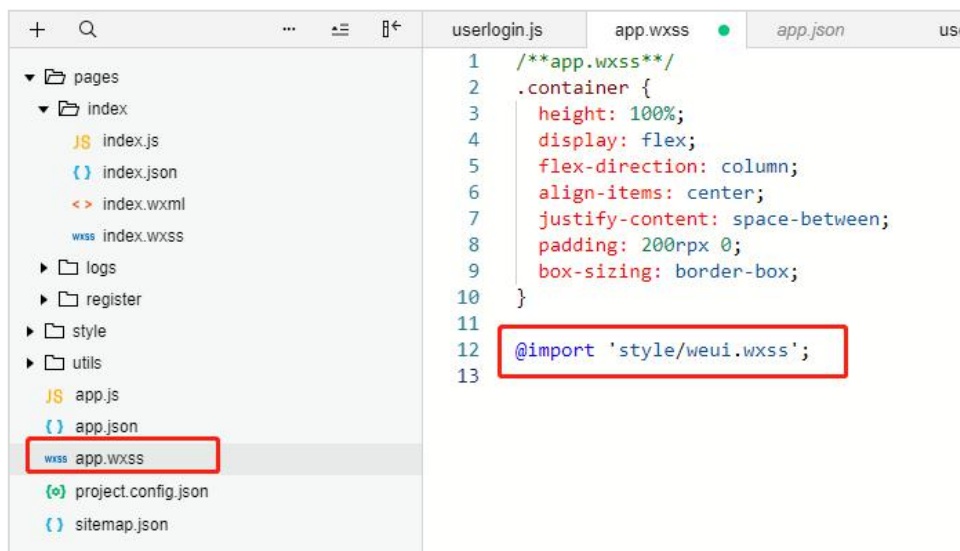


图 3 - 20 导入 weui 样式

## 2. bindchange 事件

bindchange 事件与 bindtap 事件不同，它主要是当输入框中的内容发生改变时，触发对应的事件处理函数，并且输入框中的值可以通过 event.detail.value 来获取，举个简单的例子，代码如下。

wxml 文件代码：

```
<view class="weui-cells weui-cells_after-title">
  <view class="weui-cell weui-cell_input">
    <view class="weui-cell_hd">
      <view class="weui-label">qq</view>
    </view>
```



```

    <view class="weui-cell_bd">
      <input class="weui-input" placeholder="请输入 qq"
bindchange="changevalue"/>
    </view>
  </view>
</view>

```

js 文件代码:

```

Page({
  data: {
    qq:0
  },
  changevalue:function(event){
    console.log(event)
    this.setData({
      qq: event.detail.value
    })
  },
})

```

页面效果如所示。



图 3 - 21 bindchange 使用样例

当在输入框输入内容后，将鼠标点击其他空白处，可以打印出 `changevalue` 函数的返回值，会发现输入的内容被存放在 `detail` 的 `value` 中，如图 3 - 22 所示。



图 3 - 22 bindchange 事件触发后 value 的值

### 3. openAlert 函数

openAlert 函数是在 js 文件中自定义的一个函数，在定义函数后，可以在其他函数中使用 this.openAlert()调用 openAlert 函数。

#### 3.2.2 注册页面实现

与新建 userlogin 页面一样，在 register 目录下，右击“register”，单击“新建 Page”，并命名为“register”。建完 register 页面后，接下来就是往页面里写东西了。

首先先看一下 register 页面最后的界面需要做成什么样，如图 3 - 23 所示。



图 3 - 23 注册页面效果图

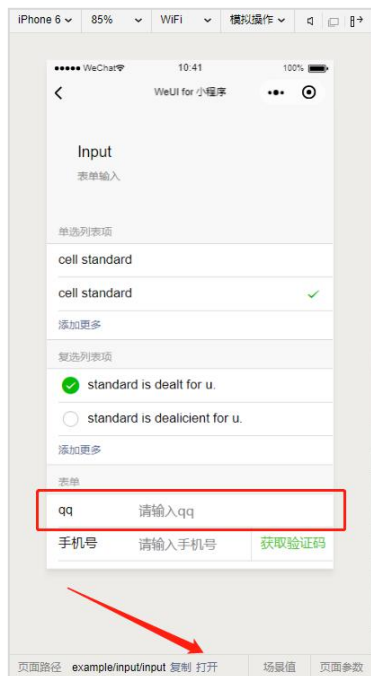


图 3 - 24 WeUI 样式库中对应的 input 样式

然后去 WeUI 基础样式库找到对应的样式，其中姓名、手机号、学校、学号和入学年份是一个输入框，对应的是 WeUI 中表单--input 里面的一种样式，如图 3 - 24 所示。单击模拟器下方“打开”按钮，即可在编辑器的目录结构区找到该页面对应的目录，打开 input.wxaml 文件，找到该样式对应的代码，如图 3 - 25 所示。将其拷贝至 doudouyun 项目的 register.wxaml，其中这段代码最后还少了一个</view>，作为最开始<view>的结束。



图 3 - 25 input.wxml 中对应样式代码

以姓名的 input 为例，其他都与姓名的操作一致，代码如下。

register.wxml 代码：

```
<view class="weui-cells weui-cells_after-title">
  <view class="weui-cell weui-cell_input">
    <view class="weui-cell_hd">
      <view class="weui-label">姓名</view>
    </view>
    <view class="weui-cell_bd">
      <input class="weui-input" placeholder="请输入姓名"
bindchange="changeName"/>
    </view>
  </view>
</view>
```

register.js 代码：

```
Page({
  data: {
    name: ''
  },
  changeName:function(e){
    this.setData({
      name: e.detail.value
    })
  }
})
```

其他注册信息的输入框与姓名一样，分别加入 wxml 代码，并在 data 数组中加入对应的变量，对应的 bindchange 函数名进行修改即可。

除了输入框外，最后还有一个提交按钮，在 WeUI 样式库中表单--button，找对应的 button 样式，如图 3 - 26 所示。



图 3 - 26 WeUI 样式库中对应的 button 样式

然后在 register.wxxml 文件的最后加上一段 button 的代码，具体代码如下。

```
<view class="page_bd page_bd_spacing submit">
  <button class="weui-btn" type="primary">提交</button>
</view>
```

其中第一个<view>的 class 类最后新加一个 submit 子类，并在 wxss 文件中 写 submit 子类样式的相关属性，主要是为了调整提交按钮的样式，其中，margin 后面如果只有两个参数的话，第一个表示 top 和 bottom，第二个表示 left 和 right，margin: 0 auto，表示上下边界为 0，左右则根据宽度自适应相同值（即居中），padding-top 作用是使 button 与 input 之间有一定距离，而不是紧紧连接在一起。并设置 width 为屏幕宽度的 90%。具体代码如下。

```
.submit{
  margin: 0 auto;
  padding-top: 15px;
  width: 90%;
}
```

提交按钮绑定的事件处理函数 bindSubmit，主要是向后台发送用户注册信息，这边后台提供了一个 Api 接口用于将注册信息存入后台数据库。请求成功后，跳转

---

至 index 页面，具体代码如下。

```
bindSubmit: function (e) {
  wx.request({
    url:
'http://zjgsujiaoxue.applinzi.com/index.php/Api/User/register_by_openid
',
    data: {
      openid: wx.getStorageSync('jiaoxue_OPENID'),
      globalData: JSON.stringify(app.globalData.userInfo),
      name: this.data.name,
      tel: this.data.tel,
      school: this.data.school,
      num: this.data.num,
      enter_year: this.data.year
    },
    success: res => {
      if (res.data.is_register) {
        wx.redirectTo({
          url: '../index/index',
        })
      }
    },
    fail: res => {
    },
  })
},
```

### 3.3 我的页面

用户在注册页面填入注册信息后，单击“提交”按钮，完成豆豆云的注册。然后就是跳转至 index 页面，这里需要新建一个我的页面，用于用户查看注册信息，本节主要讲解的就是如何开发我的页面。

#### 3.3.1 我的页面知识点讲解

我的页面主要新增了两个知识点：微信小程序媒体组件 image 的属性，WXSS 属性的介绍。

##### 1. image 属性

表 3-6 image 组件的属性

属性名	类型	说明
src	String	图片资源地址
mode	String	图片裁剪、缩放的模式
binderror	HandleEvent	当错误发生时，发布到 AppService 的事件名，事件对象 event.detail = { errMsg: 'something wrong' }
bindload	HandleEvent	当图片载入完毕时，发布到 AppService 的事件名，事件对象 event.detail = { height:'图片高度 px', width:'图片宽度 px' }

注：image 组件默认宽度 300px、高度 225px

```

1 <view class="weui-cells weui-cells_after-title">
2   <view class="weui-cell weui-cell_access" hover-class="weui-cell_active"
  bindtap='choseImage'>
3     <view class="weui-cell_bd">头像</view>
4     <view class="zan-cell__ft weui-cell__ft_in-access ">
5       <image class="head_img" src="{{userInfo.head_img?
  userInfo.head_img: '/images/default_head_circle.png'}}"></image>
6     </view>
7   </view>
8   <view class="weui-cell weui-cell_access " hover-class="weui-cell_active" |
  bindtap='bindName'>
9     <view class="weui-cell_bd ">姓名</view>
10    <view class="weui-cell__ft weui-cell__ft_in-access ">{{userInfo.name}}</view>
11  </view>
12  <view class="weui-cell weui-cell_access " hover-class="weui-cell_active"
  bindtap='bindTel'>
13    <view class="weui-cell_bd ">手机号</view>
14    <view class="weui-cell__ft weui-cell__ft_in-access ">{{userInfo.tel}}</view>
15  </view>
16  <view class="weui-cell weui-cell_access " hover-class="weui-cell_active"
  bindtap='bindSex'>
17    <view class="weui-cell_bd ">性别</view>

```

图 3-27 images 组件

图 3-27 中 image 组件用到了三目运算作为判断，三目运算符的定义：<表达式 1> ? <表达式 2> : <表达式 3>; "?" 运算符的含义是：先求表达式 1 的值，如果为真，则执行表达式 2，并返回表达式 2 的结果；如果表达式 1 的值为假，则执行表达式 3，并返回表达式 3 的结果。

试验中的 image 链接语句：src="{{userInfo.head\_img1?userInfo.head\_img:'/images/default\_head\_circle.png'}}" 是三目运算符。首先判断 storage 当中是否获取到 userInfo.head\_img。如图 3-28 所示。

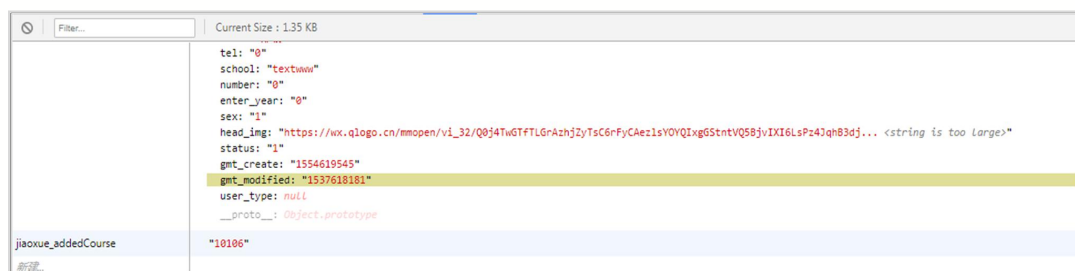


图 3-28 userinfo中头像信息

如果 storage 当中获取到 userInfo.head\_img，则打印出 userInfo.head\_img，反之则打印 image 文件中的 default\_head\_circle.png 图片。

## 2. wxss 属性介绍

rpx（responsive pixel）：可以根据屏幕宽度进行自适应。规定屏幕宽为 750rpx。如在 iPhone6 上，屏幕宽度为 375px，共有 750 个物理像素，则 750rpx = 375px = 750 物理像素，1rpx = 0.5px = 1 物理像素。

设备	rpx换算px (屏幕宽度/750)	px换算rpx (750/屏幕宽度)
iPhone5	1rpx = 0.42px	1px = 2.34rpx
iPhone6	1rpx = 0.5px	1px = 2rpx
iPhone6 Plus	1rpx = 0.552px	1px = 1.81rpx

建议：开发微信小程序时设计师可以用iPhone6作为视觉稿的标准。

注意：在较小的屏幕上不可避免的会有一些毛刺，请在开发时尽量避免这种情况。

```
.head_img {
  height: 120rpx;
  width: 120rpx;
  border-radius: 50%;
}

.weui-cell__ft {
  color: #000;
}
```

Height：图片的高度、Width：图片的宽度、border-radius：圆角的角度：为图片添加圆角边框,例如 border-radius: 50%，就是以百分比定义圆角的形状。

### 3.3.2 我的页面实现

右击“pages”，单击“新建目录”按钮，命名为“my”，右击“my”目录，单击“新建 Page”按钮。命名为“myinfo”。

我的页面的实现其实与注册页面差不多，其中我的页面的效果图，如图 3 - 29

所示。



图 3 - 29 我的页面效果图



图 3 - 30 WeUI 样式库中对应的 list 样式

首先就是在 WeUI 样式库中找到对应的样式，查看 WeUI 中 list 样式，发现要找的是带说明带跳转的列表项，如图 3 - 30 所示。myinfo.wxml 文件中的代码如下。

```
<view class="weui-cells weui-cells_after-title">
  <navigator url="" class="weui-cell weui-cell_access" hover-class="weui-cell_active">
    <view class="weui-cell__bd">头像</view>
    <view class="weui-cell__ft weui-cell__ft_in-access">
      <image class="head_img" src="{{userinfo.head_img?userinfo.head_img: '/images/default_head_circle.png'}}">
    </image>
    </view>
  </navigator>
  <navigator url="" class="weui-cell weui-cell_access" hover-class="weui-cell_active">
    <view class="weui-cell__bd">姓名</view>
    <view class="weui-cell__ft weui-cell__ft_in-access">{{userinfo.name}}</view>
  </navigator>
  <navigator url="" class="weui-cell weui-cell_access" hover-class="weui-cell_active">
    <view class="weui-cell__bd">手机号</view>
    <view class="weui-cell__ft weui-cell__ft_in-access">{{userinfo.tel}}</view>
  </navigator>
```



```

        <navigator url="" class="weui-cell weui-cell_access" hover-class="
weui-cell_active">
            <view class="weui-cell__bd">性别</view>
            <view class="weui-cell__ft weui-cell__ft_in-access">{{userinfo.s
ex}}</view>
        </navigator>
        <navigator url="" class="weui-cell weui-cell_access" hover-class="
weui-cell_active">
            <view class="weui-cell__bd">学校</view>
            <view class="weui-cell__ft weui-cell__ft_in-access">{{userinfo.s
chool}}</view>
        </navigator>
        <navigator url="" class="weui-cell weui-cell_access" hover-class="
weui-cell_active">
            <view class="weui-cell__bd">学号</view>
            <view class="weui-cell__ft weui-cell__ft_in-access">{{userinfo.n
umber}}</view>
        </navigator>
        <navigator url="" class="weui-cell weui-cell_access" hover-class="
weui-cell_active">
            <view class="weui-cell__bd">入学年份</view>
            <view class="weui-cell__ft weui-cell__ft_in-access">{{userinfo.e
nter_year}}</view>
        </navigator>
    </view>

```

其中 userinfo 的值是通过向后台访问请求，获取到的用户信息，并存在本地，然后从本地读取出来进行赋值。该请求的代码写在 app.js 中，具体代码如下。

```

wx.request({
  url:
'https://zjgsujiaoxue.applinzi.com/index.php/Api/User/getInfo',
  data: {
    'openid': res.data.openid,
  },
  success: function (res1) {
    wx.setStorageSync('userInfo', res1.data.data)
  },
})

```

其中 myinfo.js 文件的 data 数组中定义 userinfo，并在 onLoad 函数中对 userinfo 变量进行赋值，具有代码如下

```

Page({

  /**

```

```

* 页面的初始数据
*/
data: {
  userinfo: { }
},

/**
* 生命周期函数--监听页面加载
*/
onLoad: function (options) {
  this.setData({
    userinfo: wx.getStorageSync('userInfo')
  })
}

```

编译后发现，头像显示过大，如图 3 - 31 所示。



图 3 - 31 头像显示过大

因此需要对媒体组件 image 中自定义类“head\_img”，调整图片大小，其中 myinfo.wxss 文件的代码如下。

```

.head_img{
  height: 120rpx;
  width: 120rpx;
  border-radius: 50%;
}

```

到这里，我的页面就能正常显示了。