

微信应用号开发教程

本文来源于 <https://my.oschina.net/wwnick/blog/750055>，权利及责任归“博卡君”。

本文档将带你一步步创建完成一个微信小程序，并可以在手机上体验该小程序的实际效果。这个小程序的首页将会显示欢迎语以及当前用户的微信头像，点击头像，可以在新开的页面中查看当前小程序的启动日志。

1. 获取微信小程序的 AppID

首先，我们需要拥有一个帐号，如果你能看到该文档，我们应当已经邀请并为你创建好一个帐号。注意不可直接使用服务号或订阅号 AppID。利用提供的帐号，登录 <https://mp.weixin.qq.com>，就可以在网站的「设置」-「开发者设置」中，查看到微信小程序的 AppID 了。



首页



开发管理



用户身份



数据分析



模板消息



设置

设置

基本设置

开发设置

开发者ID

AppID(小程序ID)

AppSecret(小程序密钥)

注意：如果我们不是用注册时绑定的管理员微信号，在手机上体验该小程序。那么我们还需要操作「绑定开发者」。即在「用户身份 - 开发者」模块，绑定上需要体验该小程序的微信号。本教程默认注册帐号、体验都是使用管理员微信号。

2. 创建项目

我们需要通过开发者工具，来完成小程序创建和代码编辑。

开发者工具安装完成后，打开并使用微信扫码登录。选择创建「项目」，填入上文获取到的 AppID，设置一个本地项目的名称（非小程序名称），比如「我的第一个项目」，并选择一个本地的文件夹作为代码存储的目录，点击「新建项目」就可以了。

为方便初学者了解微信小程序的基本代码结构，在创建过程中，如果选择的本地文件夹是个空文件夹，开发者工具会提示，是否需要创建一个 quick start 项目。选择「是」，开发者工具会帮助我们在开发目录里生成一个简单的 demo。

< 返回

新建项目

AppID

填写小程序AppID，可在公众平台开发设置页中查看

项目名称

本地开发目录

选择

取消

添加项目

项目创建成功后，我们就可以点击该项目，进入并看到完整的开发者工具界面，点击左侧导航，在「编辑」里可以查看和编辑我们的代码，在「调试」里可以测试代码并模拟小程序在微信客户端效果，在「项目」里可以发送到手机里预览实际效果。

3. 编写代码

点击开发者工具左侧导航的「编辑」，我们可以看到这个项目，已经初始化并包含了一些简单的代码文件。最关键也是必不可少的，是 app.js、app.json、app.wxss 这三个。其中，.js 后缀的是脚本文件，.json 后缀的文件是配置文件，.wxss 后缀的是样式表文件。微信小程序会读取这些文件，并生成小程序实例。

下面我们简单了解这三个文件的功能，方便修改以及从头开发自己的微信小程序。

app.js 是小程序的脚本代码。我们可以在这个文件中监听并处理小程序的生命周期函数、声明全局变量。调用 MINA 提供的丰富的 API，如本例的同步存储及同步读取本地数据。

```
//app.js App({  onLaunch: function () {      // 调用 API 从本地缓存中获取数据      var logs = wx.getStorageSync('logs') || []      logs.unshift(Date.now())      wx.setStorageSync('logs', logs)  },  getUserInfo:function(cb){      var that = this;      if(this.globalData.userInfo){          typeof cb == "function" &&          cb(this.globalData.userInfo)      }else{          // 调用登录接口          wx.login({              success: function ()                  that.globalData.userInfo = res.userInfo;          },          type: 'success'          }, {              success: function (res) {                  that.globalData.userInfo = res.userInfo;              },              fail: function () {}          },          cb(that.globalData.userInfo)      })      });  },  globalData:{      userInfo:null  } })
```

app.json 是对整个小程序的全局配置。我们可以在这个文件中配置小程序是由哪些页面组成，配置小程序的窗口 背景色，配置导航条样式，配置默认标题。注意该文件不可添加任何注释。

```
/**app.json*/
{  "pages":[    "pages/index/index",    "pages/logs/logs"  ],  "window":{    "backgroundTextStyle":"light",    "navigationBarBackgroundColor": "#fff",    "navigationBarTitleText":    "WeChat",    "navigationBarTextStyle":"black"  } }
```

app.wxss 是整个小程序的公共样式表。我们可以在页面组件的 class 属性上直接使用 app.wxss 中声明的样式规则。

```
/**app.wxss**/ .container {  height: 100%;  display: flex;  flex-direction: column;  align-items: center;  justify-content: space-between;  padding: 200rpx 0;  box-sizing: border-box; }
```

3. 创建页面

在这个教程里，我们有两个页面，index 页面和 logs 页面，即欢迎页和小程序启动日志的展示页，他们都在 pages 目录下。微信小程序中的每一个页面的【路径 + 页面名】都需要写在 app.json 的 pages 中，且 pages 中的第一个页面是小程序的首页。

每一个小程序页面是由同路径下同名的四个不同后缀文件的组成，如：index.js、index.wxml、index.wxss、index.json。 .js 后缀的文件是脚本文件，.json 后缀的文件是配置文件，.wxss 后缀的是样式表文件，.wxml 后缀的文件是页面结构文件。

index.wxml 是页面的结构文件：

```

<!--index.wxml--> <view class="container">  <view bindtap="bindViewTap" class="userinfo">      <image
class="userinfo-avatar" src="{{userInfo.avatarUrl}}" background-size="cover"></image>      <text
class="userinfo-nickname">{{userInfo.nickName}}</text>    </view>    <view class="usermotto">      <text
class="user-motto">{{motto}}</text>    </view> </view>

```

本例中使用了 <view/>、<image/>、<text/> 来搭建页面结构，绑定数据和交互处理函数。

index.js 是页面的脚本文件，在这个文件中我们可以监听并处理页面的生命周期函数、获取小程序实例，声明并处理数据，响应页面交互事件等。

```

//index.js // 获取应用实例 var app = getApp() Page({  data: {      motto: 'Hello World',      userInfo: {}  },  //
事件处理函数  bindViewTap: function() {      wx.navigateTo({          url: '../logs/logs'      })  },  onLoad: function
() {      console.log('onLoad')      var that = this      // 调用应用实例的方法获取全局数
据      app.getUserInfo(function(userInfo){          // 更新数
据      that.setData({          userInfo:userInfo      })      })  } })

```

index.wxss 是页面的样式表：

```

/**index.wxss**/ .userinfo{  display: flex;  flex-direction: column;  align-items: center; } .userinfo-avatar
{  width: 128rpx;  height: 128rpx;  margin: 20rpx;  border-radius: 50%; } .userinfo-nickname {  color:
#aaa; } .usermotto {  margin-top: 200px; }

```

页面的样式表是非必要的。当有页面样式表时，页面的样式表中的样式规则会层叠覆盖 app.wxss 中的样式规则。如果不指定页面的样式表，也可以在页面的结构文件中直接使用 app.wxss 中指定的样式规则。

index.json 是页面的配置文件：

页面的配置文件是非必要的。当有页面的配置文件时，配置项在该页面会覆盖 app.json 的 window 中相同的配置项。如果没有指定的页面配置文件，则在该页面直接使用 app.json 中的默认配置。

logs 的页面结构

```
<!--logs.wxml--> <view class="container log-list"> <block wx:for-items="{{logs}}" wx:for-item="log"> <text  
class="log-item">{{index + 1}}. {{log}}</text> </block> </view>
```

logs 页面使用 <block/> 控制标签来组织代码，在 <block/> 上使用 wx:for-items 绑定 logs 数据，并将 logs 数据循环展开节点

```
//logs.js var util = require('../..//utils/util.js') Page({ data: { logs: [] }, onLoad: function ()  
{ this.setData({ logs: (wx.getStorageSync('logs') || []).map(function (log) { return  
util.formatTime(new Date(log)) }) }) }) })
```

运行结果如下：

设置 动作 帮助

微信开发者工具 0.9.091800



iPhone 6

wifi



Console

Sources

Network



top



Preserve log



编辑



调试



项目



重启



后台

WeChat

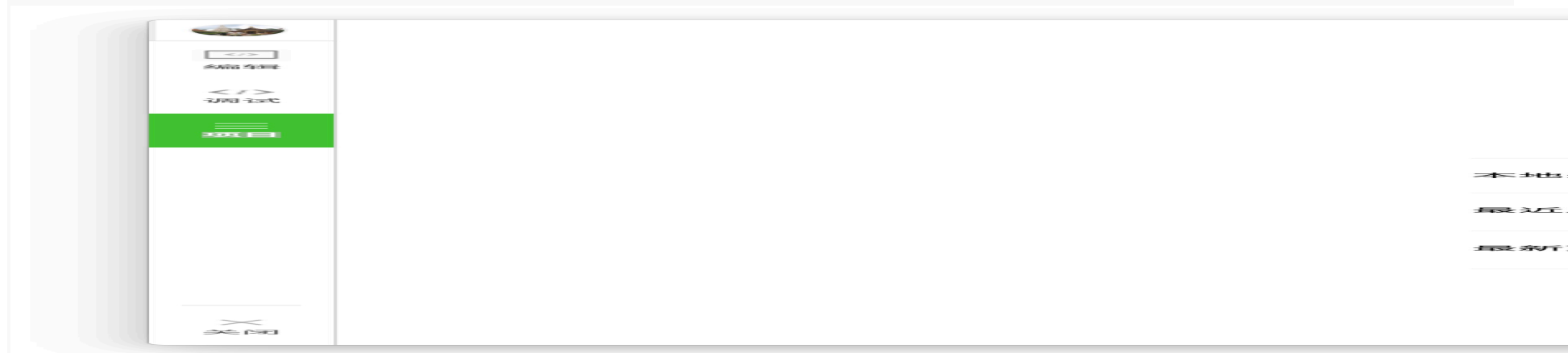


开发小能手

onLoad



4. 手机预览



开发者工具左侧菜单栏选择「项目」，点击「预览」，扫码后即可在微信客户端中体验。

目前，预览和上传功能尚无法实现，需要等待微信官方的下一步更新。

如你所见，微信官方给出的开发指南还非常简单，很多细节、代码和功能都没有明确的展示，所以接下来就到博卡君展示实力的时候啦！开发教程正式开始！

第一章：准备工作

做好准备工作很重要。开发一个微信应用号，你需要提前到微信的官方网站（weixin.qq.com）下载开发者工具。

1. 下载最新微信开发者工具，打开后你会看到该界面：



请选择项目



添加项目



cards



wxapp



cards



course

2. 点击「新建 web+」项目，随后出现如下画面：

< 返回

新建项目

AppID

cards

填写分配给大家的 APPID

Appname

cards

本地开发目录

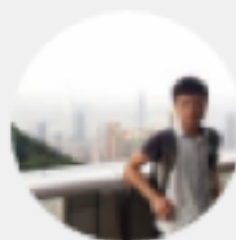
E:\workspace\card_new\bizzz

3. 该页面内的各项内容需要注意——

- AppID: 依照官方解释来填。
- Appname: 项目最外层文件夹名称，如你将其命名为「ABC」，则之后的全部项目内容均将保存在「/ABC/...」目录下。
- 本地开发目录: 项目存放在本地的目录。

注：再次强调，如果你和团队成员共同开发该项目，则建议你们使用同样的目录名称及本地目录，以确保协同开发的统一性。如果你之前已有项目，则导入过程与以上内容近似，不再赘述。

4. 准备工作全部完成后，点击「新建项目」按钮，弹出框点「确定」。



请选择项目



新建项目



cards

5. 如上图所示，此刻，微信开发者工具已经为你自动构建了一个初始的 demo 项目，该项目内包含了一个微信应用项目所需具备的基本内容和框架结构。点击项目名称（图中即「cards」）进入该项目，就能看到整个项目的基本架构了：



iPhone 6

with

调试



编辑



调试



项目



重启



后台

Web+

接口

Navigator 跳转

UI 界面

DataStorage 数据存储

Account 账号

Networking 网络

Media 媒体

Location 地理位置

Device 设备

Console Sources Network Storage AppData

top Preserve log

```
App: onLaunch have been invoked
App Launch
App: onShow have been invoked
App Show
Register Page: index
Register Page: API/media
Register Page: components
Register Page: apps
Register Page: API/navigator
Register Page: API/ui
Register Page: API/data-storage
Register Page: API/account
Register Page: API/networking
Register Page: API/location
Register Page: API/device
Register Page: API/data
Register Page: API/address
Register Page: API/pay
Register Page: API/share
Register Page: API/others
Register Page: components/wx-canvas
Register Page: apps/animation/index
Register Page: apps/reddit/index
On app route: index
Index: onLoad have been invoked
Index onLoad
Index: onShow have been invoked
Update view with init data
Object (APIList: Array(13), __webviewId__: 0)
Index: onReady have been invoked
```

第二章：项目构架

微信目前用户群体非常庞大，微信推出公众号以后，火爆程度大家都看得到，也同样推动着 h5 的高速发展，随着公众号业务的需求越来越复杂，应用号现在的到来也是恰到好处。我们团队具体看了一两次文档后发现，它提供给开发者的方式也在发生全面的改变，从操作 DOM 转为操作数据，基于微信提供的一个过桥工具实现很多 h5 在公众号很难实现的功能，有点类似于 hybrid 开发，不同于 hybrid 开发的方式是：微信开放的接口更为严谨，结构必须采用他提供给我们的组件，外部的框架和插件都不能在这里使用上，让开发者完全脱离操作 DOM，开发思想转变很大。

工欲善其事，必先利其器。理解它的核心功能非常重要，先了解它的整个运作流程。

生命周期：

在 index.js 里面：

```
},  
onMenuShareTimeline: function() {  
    return {  
        title: "首页"  
    }  
},  
onLoad: function(options){  
    // 页面初始化 options为页面跳转所带来的参数  
    console.log("index-----inl  
},  
onReady: function(){  
    // 页面渲染完成  
    console.log("index-----on  
},  
onShow: function(){  
    // 页面显示
```

开发者工具上 Console 可以看到：

Console Sources Network Storage AppData



top



Preserve log

Filter



Regex



Hide network messages

All

App: onLaunch have been invoked

App Launch

App: onShow have been invoked

App Show

Register Page: index

Register Page: API/media

Register Page: components

Register Page: apps

Register Page: API/navigator

Register Page: API/ui

Register Page: API/data-storage

Register Page: API/account

在首页 console 可以看出顺序是 App Launch-->App Show-->onload-->onShow-->onReady。

首先是整个 app 的启动与显示，app 的启动在 app.js 里面可以配置，其次再进入到各个页面的加载显示等等。

可以想象到这里可以处理很多东西了，如加载框之类的都可以实现等等。

路由：

路由在项目开发中一直是个核心点，在这里其实微信对路由的介绍很少，可见微信在路由方面经过很好的封装，也提供三个跳转方法。

wx.navigateTo(OBJECT)：保留当前页面，跳转到应用内的某个页面，使用 wx.navigateBack 可以返回到原页面。

wx.redirectTo(OBJECT)：关闭当前页面，跳转到应用内的某个页面。

wx.navigateBack()：关闭当前页面，回退前一页面。

这三个基本上使用足够，在路由方面微信封装的很好，开发者根本不用去配置路由，往往很多框架在路由方面配置很繁琐。

组件：

此次微信在组件提供方面也是非常全面，基本上满足项目需求，故而开发速度非常快，开发前可以认真浏览几次，开发效率会很好。

其它：

任何外部框架以及插件基本上无法使用，就算原生的 js 插件也很难使用，因为以前我们的 js 插件也基本上全部是一操作 dom 的形式存在，而微信应用号此次的架构是不允许操作任何 dom，就连以前我们习惯使用的动态设置的 rem.js 也是不支持的。

此次微信还提供了 WebSocket，就可以直接利用它做聊天，可以开发的空间非常大。

跟公众号对比我们发现，开发应用号组件化，结构化，多样化。新大陆总是充满着惊喜，更多的彩蛋等着大家来发现。

接下来开始搞一些简单的代码了！

1. 找到项目文件夹，导入你的编辑器里面。在这里，我使用了 Sublime Text 编辑器。你可以根据自己的开发习惯选择自己喜欢的编辑器。

E:\workspace\card_course\index.wxml (card_course) - Sublime Text

文件(F) 编辑(E) 选择(S) 查找(I) 查看(V) 转到(G) 工具(T) 项目(P) 首选项(N) 帮助(H)

打开文件

目录

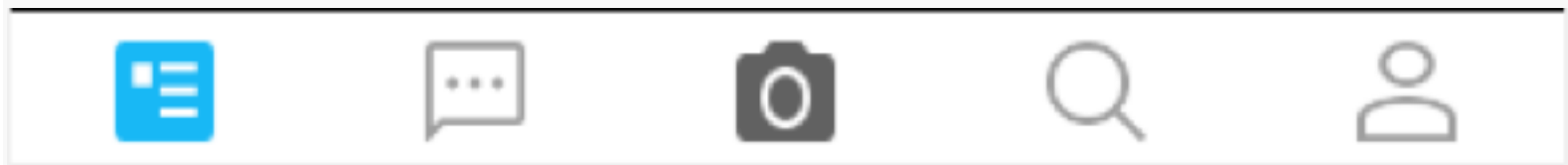
- ▼ card_course
 - ▶ API
 - ▶ apps
 - ▶ components
 - ▶ css
 - ▶ images
 - app.css
 - app.js
 - app.json
 - app.wxss
 - apps.js
 - apps.wxml
 - components.js
 - components.wxml
 - index.js
 - index.wxml
 - log.wxml

index.wxml

```
1 <wx-view>
2   <wx-view class="page-title">接口</wx-view>
3
4   <wx-view>
5     <wx-repeat items="{{APIList}}" item="API">
6       <wx-navigator url="{{API.url}}">
7         <wx-button type="primary">{{API.name}}<
8       </wx-navigator>
9     </wx-repeat>
10  </wx-view>
11 </wx-view>
12
```

2. 接下来，你需要根据自己的项目内容调整项目结构。在范例项目中，「card_course」目录下面主要包含了「tabBar」页面以及该应用的一些配置文件。

3. 示例项目的「tabBar」是五个菜单按钮：



4. 找到「app.json」文件，用来配置这个五个菜单。在代码行中找到「" tabBar"」：

```
8 },
9 "tabBar": {
10   "color": "#dddddd",
11   "selectedColor": "#3cc51f",
12   "borderStyle": "black",
13   "backgroundColor": "#ffffff",
14   "list": [{
15     "pagePath": "index",
16     "iconPath": "images/wechat.png",
17     "selectedIconPath": "images/wechatHL.png",
18     "text": "接口"
19   }, {
20     "pagePath": "components",
21     "iconPath": "images/wechat.png",
22     "selectedIconPath": "images/wechatHL.png",
23     "text": "组件"
24   }]
```












你可以根据实际项目需求更改，其中：

- 「Color」是底部字体颜色，「selectedColor」是切换到该页面高亮颜色，「borderStyle」是切换菜单上面的一条线的颜色，「backgroundColor」是底部菜单栏背景颜色。文字描述较为抽象，建议你一一调试并查看其效果，加深印象。
- 「“list”」下的代码顺序必须依次放置，不能随便更改。
- 「”pagePath”」之后的文件名内，「.wxml」后缀被隐藏起来了，这是微信开发代码中人性化的一点——帮你节约写代码的时间，无须频繁声明文件后缀。
- 「”iconPath”」为未获得显示页面的图标路径，这两个路径可以直接是网络图标。
- 「”selectedIconPath”」为当前显示页面高亮图标路径，可以去掉，去掉之后会默认显示为「”iconPath”」的图标。
- 「”Text”」为页面标题，也可以去掉，去掉之后纯显示图标，如只去掉其中一个，该位置会被占用。

注意：微信的底部菜单最多支持五栏（五个 icons），所以在设计微信应用的 UI 和基本架构时就要预先考虑好菜单栏的排布。

5. 根据以上代码规则，我做好了示例项目的基本架构，供你参考：

```
},
"tabBar": {
  "color": "#dddddd",
  "selectedColor": "#3cc51f",
  "borderStyle": "black",
  "backgroundColor": "#ffffff",
  "list": [{
    "pagePath": "index",
    "iconPath": "images/tabBar/my-card.png",
    "selectedIconPath": "images/tabBar/my-card-acti
  }, {
    "pagePath": "message_center",
    "iconPath": "images/tabBar/messges.png",
    "selectedIconPath": "images/tabBar/messges-acti
  }, {
    "pagePath": "offline_card_ocr",
    "iconPath": "images/tabBar/placeholder.png"
```

- ▶  demo
- ▶  images
- ▶  page
- ▶  service
- ▶  wxss
 - app.js
 - app.json
 -  app.wxss
 - center.js
 -  center.wxml
 -  center.wxss
 - index.js
 -  index.wxml
 - message_center.js
 -  message_center.wxml
 -  message_center.wxss
 - offline_card OCR.js

6. 「Json」文件配置好后，「card_course」的基本结构入上图所示，不需要的子集都可以暂时删除，缺少的子集则需要你主动新建。删除子集时记得顺带检查一下「app.json」里的相关内容是否已经一并删除。

注意：我个人建议你新建一个「wxml」文件的同时，把对应的「js」和「wxss」文件一起新建好，因为微信应用号的配置特点就是解析到一个「wxml」文件时，会同时在同级目录下找到同文件名的「js」和「wxss」文件，所以「js」文件需及时在「app.json」里预先配置好。

编写「wxml」时，根据微信应用号提供的接口编码即可，大部分就是以前的「div」，而我们现在就用「view」即可。需要用其它子集时，可以根据微信提供的接口酌情选择。

使用「class」名来设置样式，「id」名在这里基本没有什么用处。主要操作数据，不操作「dom」。

```

<view>
  <scroll-view class="header_wrap">
    <view class="header">
      <searchbar bindchange="bindInputChange" />
    </view>
    <view class="menu" bindtap="bindButtonTapSheet">=</view>

    <view class="new_people" bindtap="bindClickCardGroup">新的人脉</view>
    <view class="divide">分组</view>

    <view class="sort_list_wrap" style="top:{{indexTop}}">
      <view class="sort_list_A" bindtap="lettersTap" id="sort_list_A">{{letters}}</view>
      <block wx:for-items="{{datauser1}}" wx:for-item="msg" wx:for-index="idx">
        <navigator url="page/card/card_detail">
          <drawer class="user_card" style="display:{{display}}" first-drawerText="删除" second-drawer-text="拨号" second
            bindtap="bindDrawerTap" id="{{idx}}">
            <image class="userphoto" src="{{msg.thumbnail}}"/>
            <view class="userinfo" id="userinfo">
              <view class="username">{{msg.userName}}</view>
              <view class="userjob">{{msg.unit}}<text class="userpost">{{msg.job}}</text></view>
              <view class="useraddress">{{msg.company}}</view>
            </view>
          </drawer>
        </navigator>
      </block>
    </view>
    <view class="no_seach {{datauser}}">抱歉，无法为您搜索到相关的数据</view>
    <view class="seach_wrap">
      <view class="right_seach">
        <block wx:for-items="{{sort}}" wx:for-item="sortmsg">
          <view class="sort_list" bindtap="sortTap">{{sortmsg}}</view>
        </block>
      </view>
    </view>
  </scroll-view>
</view>
<view class="index_menu_wrap" style="display: {{indexMenuDisplay}};">
  <view class="index_menu">
    <view class="index_menu_borderht">

```


7. 以上是示例项目首页的「wxml」编码。从图中就可以看出，实现一个页面代码量非常少。

8. 「Wxss」文件是引入的样式文件，你也可以直接在里面写样式，示例中采用的是引入方式：



```
1 @import "wxss/index.css";
2 body{
3     //测试使用，支持直接编写css样式
4     background:pink;
5 }
```

设置 动作 帮助

微信开发者工具 0.9.091800

iPhone 6

wifi

调试

名片盒

搜索

新的人脉

分组

H

黄海强

市场部 经理

中国电信广东公司

黄海强

市场部 经理

平安证券有限责任公司

黄海强

市场部 经理

中国电信股份有限公司深圳分公司

黄海强

重庆安然文化传播有限责任公司

NORGIN

上海诺安信息科技股份有限公司

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Elements

Console

Sources

Network

Timeline

Profiles

Application

Storage

```
<!DOCTYPE html>
<html lang="zh-CN">
  <head>_</head>
  <body style="">
    <div id="container">_</div>
    <script>_</script>
    <script>_</script>
  </body>
</html>
```

html

body

Console

top

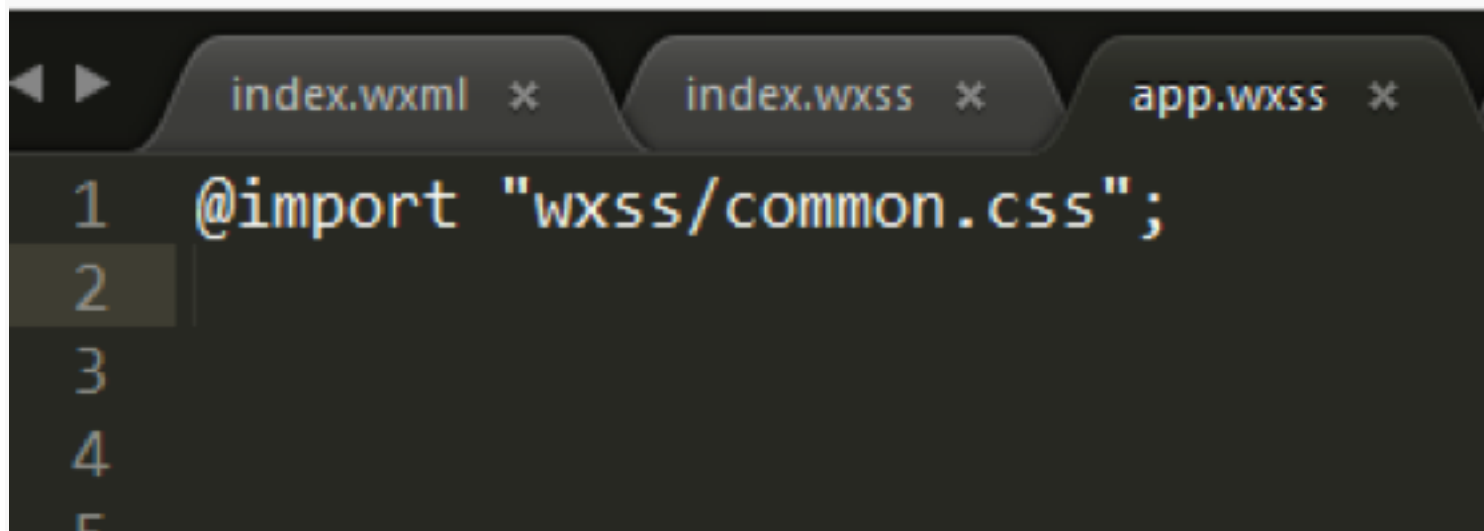
Preserve log

图片组件: 当前环境是开发者工具或android

9. 修改代码后刷新一次，可以看到未设背景的「view」标签直接变成了粉色。

注意：修改「wxml」和「wxss」下的内容后，直接 F5 刷新就能直接看到效果，修改「js」则需点击重启按钮才能看到效果。

10. 另外，公共样式可以在「app.wxss」里直接引用。

A screenshot of a code editor interface with a dark theme. At the top, there are three tabs: 'index.wxml', 'index.wxss', and 'app.wxss'. The 'app.wxss' tab is active. The code editor shows the following content:

```
1  @import "wxss/common.css";  
2  
3  
4  
5
```

The line numbers 1 through 5 are visible on the left side of the editor. The code is written in a light blue/cyan color on the dark background.

index.wxml ×

app.json ×

index.wxss ×

×

```
1 {  
2   "pages": [  
3     "index",  
4     "message_center",  
5     "offline_card_ocr",  
6     "seach_company",  
7     "center",  
8     "page/card/card_detail",  
9     "page/card/card_album_list",  
0     "page/card/card_edit",  
1     "page/card/card_edit_2"  
2   ],
```

11. 「Js」文件需要在「app.json」文件的「" page" 」里预先配置好。为了项目结构清晰化，我在示例项目中的「index」首页同级目录新建其它四个页面文件，具体如下：

- ▶ demo
- ▶ images
- ▶ page
- ▶ service
- ▶ wxss
 - app.js
 - app.json
 - app.wxss
 - center.js
 - center.wxml
 - center.wxss
 - index.js
 - index.wxml
 - index.wxss
 - message_center.js
 - message_center.wxml
 - message_center.wxss

经过以上步骤，案例中的五个底部菜单就全部配置完毕了。