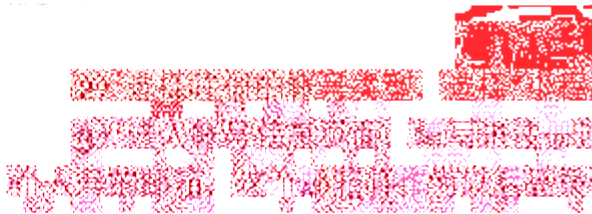
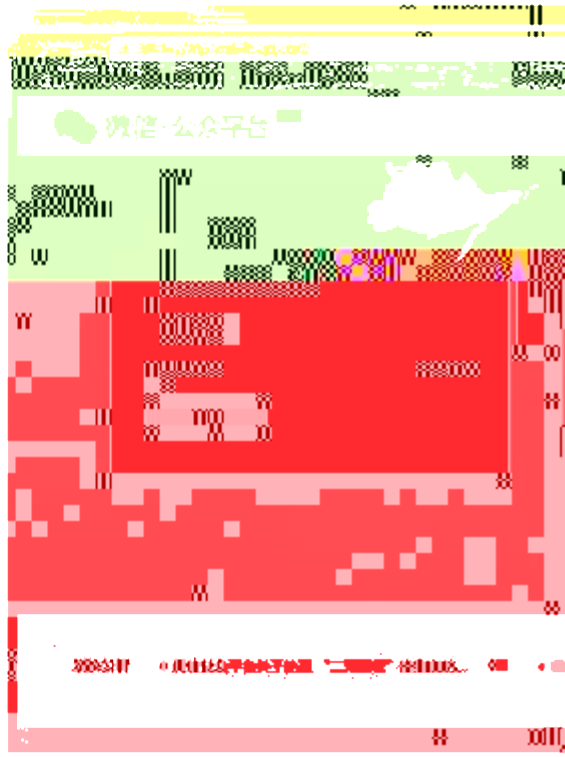


长汀一中选修2课程

自己动手做一个微信小程序

1.2.1 注册小程序帐号

注册小程序帐号只需如下四步：





微信公众平台

微信公众平台-小程序介绍

https://mp.weixin.qq.com/wxopen/wacontractorpage?action=step3&lang=zh_CN&token=6552265

应用 | email | 品牌管理 | 图文文章 | 开发指南 | 指南 | 其他 | 商家网站 | 新平台 | 书籍 | 帮助 | 小程序

请确认你的微信公众平台主体类型属于政府、媒体、企业、其他组织，并请按照对应的类别进行信息登记。
点击查看微信公众平台信息登记指引。

主体类型 如何选择主体类型？

企业	政府	媒体	其他组织
----	----	----	------

企业包括：企业、分支机构、个体工商户、企业相关品牌。

主体信息登记

企业类型 企业 个体工商户

企业名称

统一社会信用代码

组织机构代码

注册地址

所属行业

资质其他信息登记





5678

https://mp.weixin.qq.com/haspoe/admin?action=getjs-wxmpact/chan=20130527148?arg=zh_CN

微信公众平台 | 小程序

文档

首页

开发指南

小程序

开发案例

常见问题

联系我们

用户身份

管理员

开发者

运营者

开发者

小程序ID

名称

已验证

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

已验证

小程序ID

名称

类型

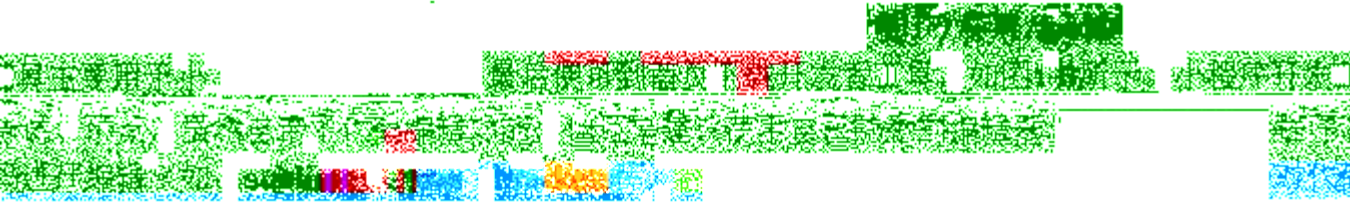
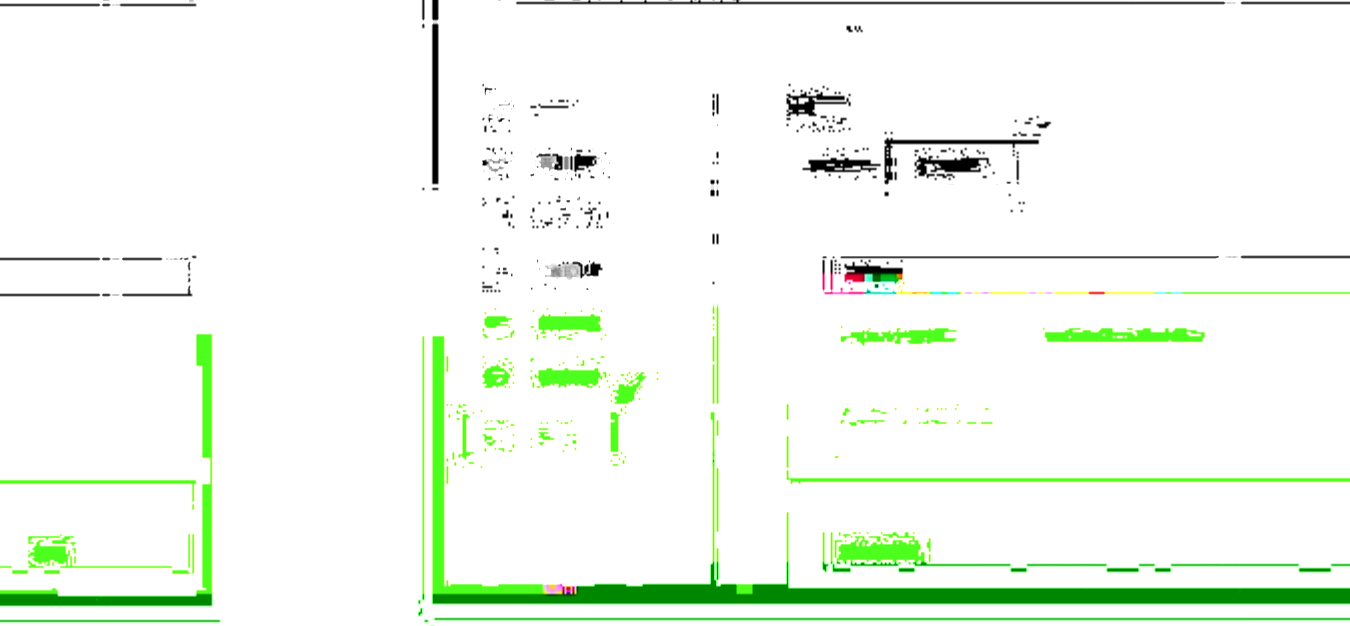
已验证

小程序ID

名称

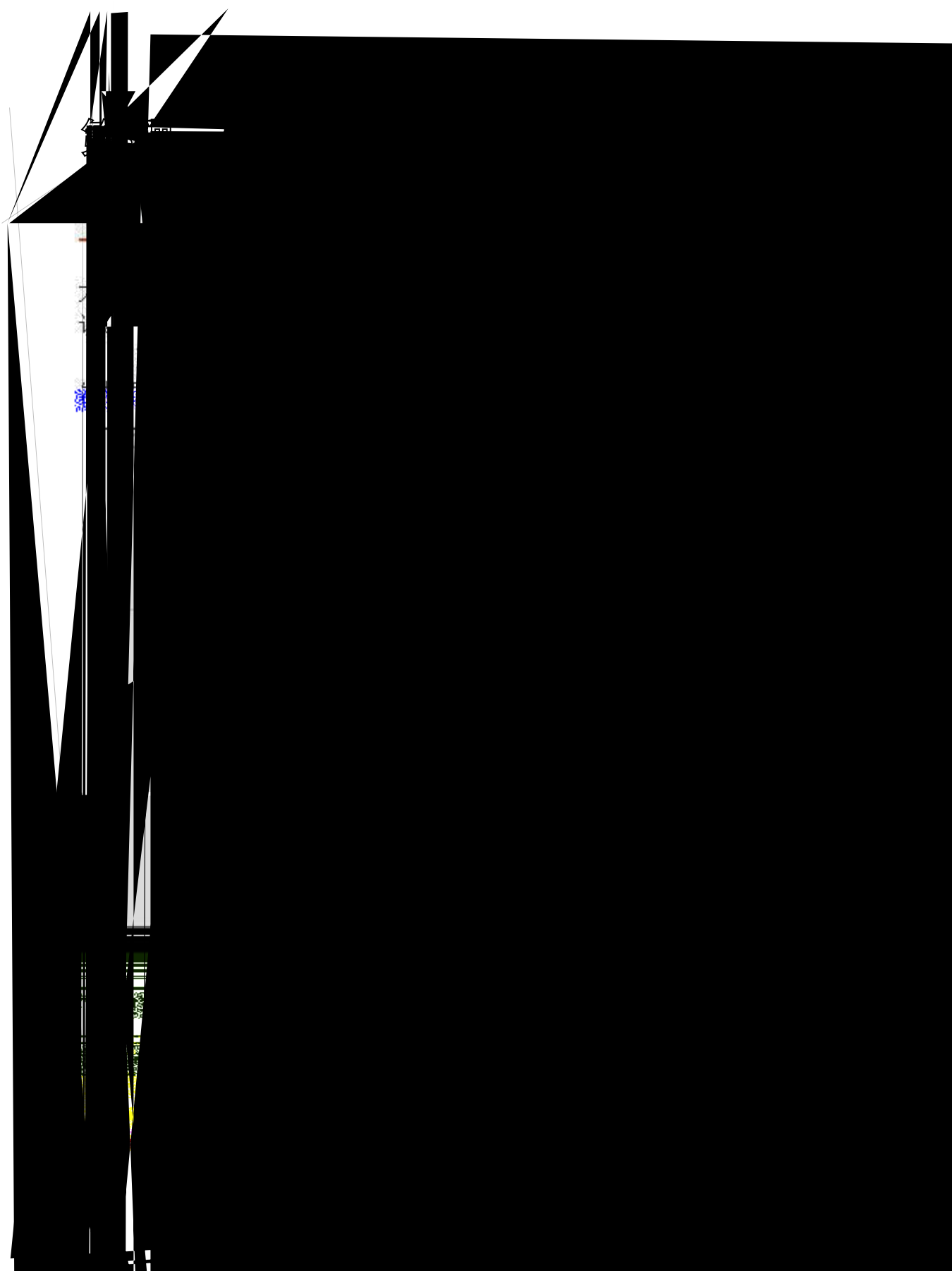
类型

已验证





小程序发布流程



< 返回

添加项目

AppID 无 AppID 部分功能受限

返回填写小程序AppID

项目名称 demo

项目目录 E:\weixin\demo

选择

在当前目录中创建 quick start 项目

取消



```
<!--index.wxml-->
```

```
<view class="container">
```

```
  <view bindtap="bindViewTap" class="userinfo">
```

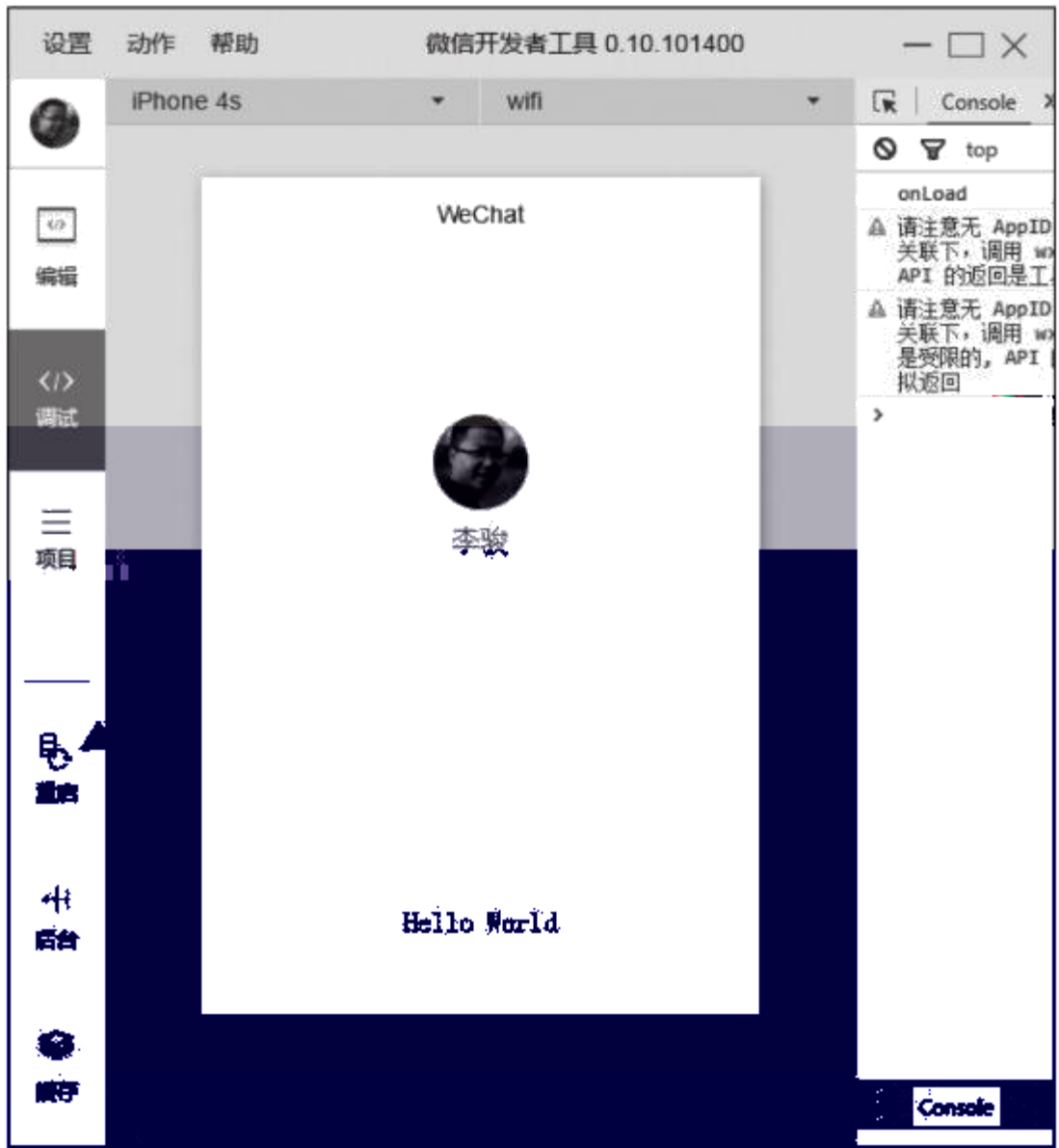
serInfo.avatarUrlserInfo.nickName

```
</view>
```

```
  <view class="usermotto">
```

```
    <!-- 修改这句代码 -->
```

```
    <text class="user-motto">我的第一个小程序</text>
```







编辑



调试

项目



本地开发目录 E:\work\wxapp-book\书籍\3 第三章 布局\demo

微信上传时间 未上传

上传

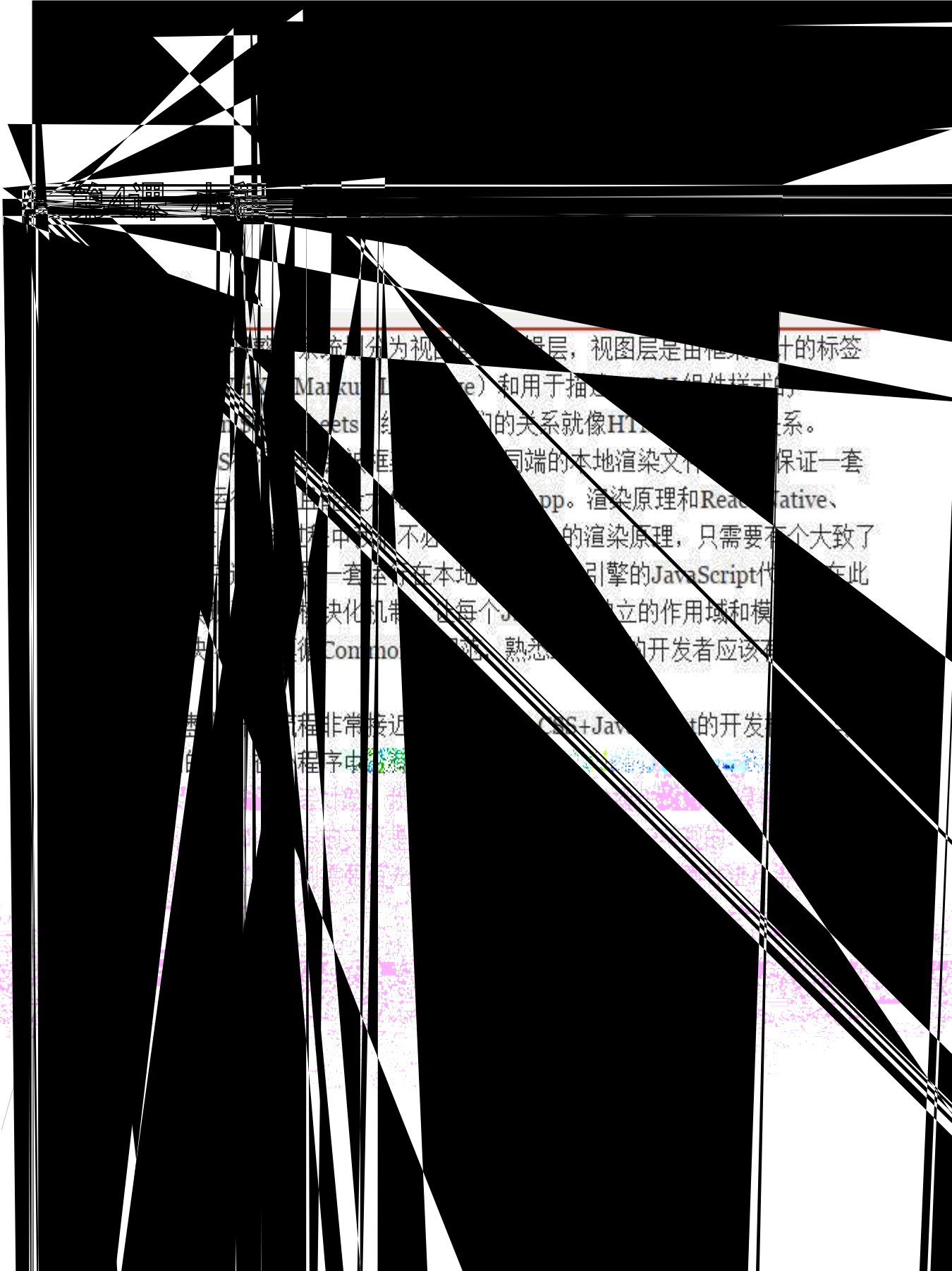
微信预览时间 2016/11/18 上午11:27:36, 微信预览大小 59 kb

预览

开启 CSS 类 BOM

监听文件变化, 自动刷新开发者工具

开启代码压缩上传



视图层

视图层分为视图层和表现层，视图层是由框架设计的标签（Markup Language）和用于描述UI组件样式的CSS（Cascading Style Sheets）组成的。它们的关系就像HTML和CSS的关系。视图层框架保证一套同端的本地渲染文件，保证一套运行在客户端的App。渲染原理和React Native、Flutter等框架中并不一致，渲染原理，只需要有个大致了解即可。视图层是一套运行在本地渲染引擎的JavaScript代码。在此过程中，模块化机制，每个UI组件独立的作用域和模块，使得CommonJS等模块化机制，熟悉的开发者应该可以很容易地理解。视图层非常接近于传统的CSS+JavaScript的开发模式，在视图层程序中，

视图层

用户交互行为，通过 WXML 事件绑定，调用逻辑层相关事件方法。

逻辑层调用 setData() 触发视图层渲染。

逻辑层

视图层与逻辑层交互

视图层与逻辑层交互



视图层

视图层与逻辑层交互

视图层与逻辑层交互



视图层

视图层与逻辑层交互

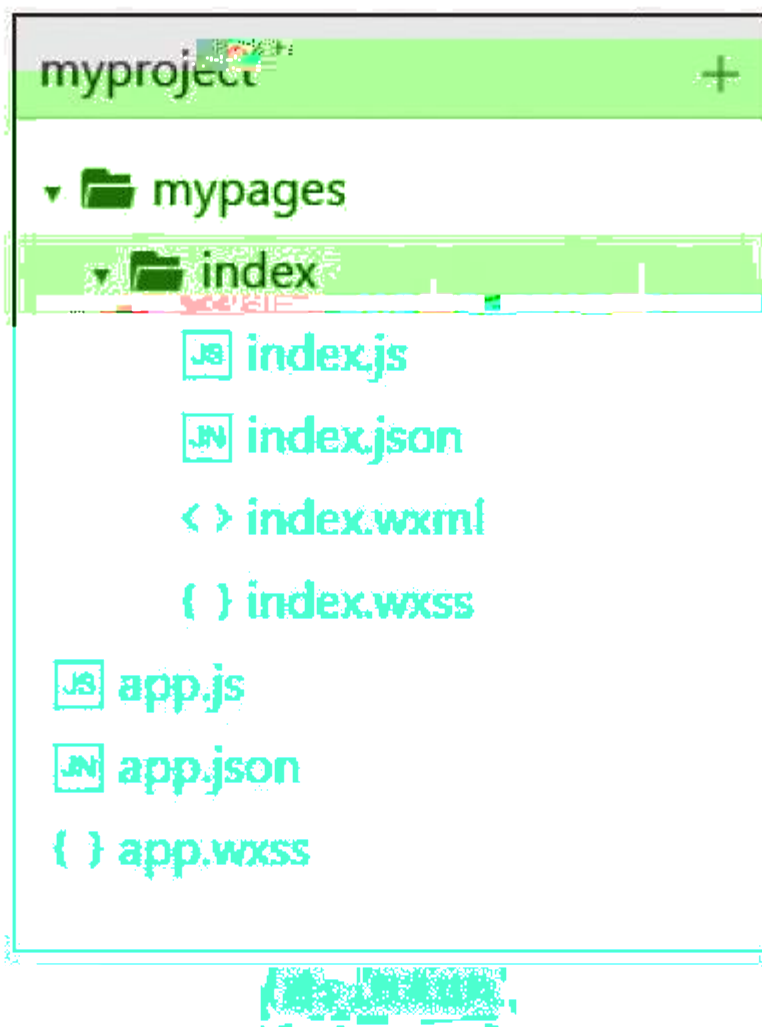
```
<view bindtap="countClick">我是index页面,你点击了{{count}}次</view>
```



经过这几步，一个简单的小程序就搭建好了。返回电脑端，刷新页面，这个小程序就部署好了。现在可以测试了，测试方法很简单，就是在微信中查看运行小程序。

④ 打开微信开发调试工具，输入小程序ID和小程序名称，点击“新增”按钮添加。





< 返回

添加项目

AppID 无 AppID 部分功能受限

返回填写小程序AppID

项目名称 myproject

项目目录 E:\weixin\myproject

选择

取消 添加项目

图2-3 项目配置界面

7) 导入项目后我们便能看到运行界面，当我们点击文字时，点击次数也会统计

设置 动作 帮助

微信开发者工具 0.10.101400



iPhone 4



wifi

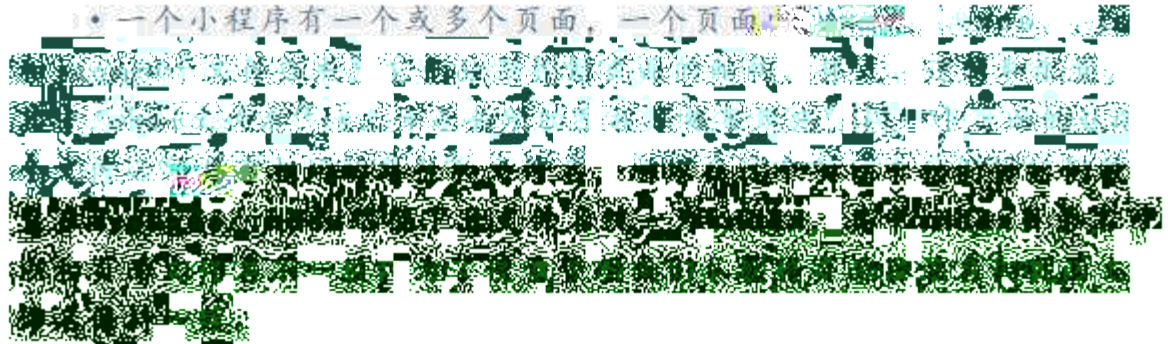


Console

小程序在 iOS 上运行

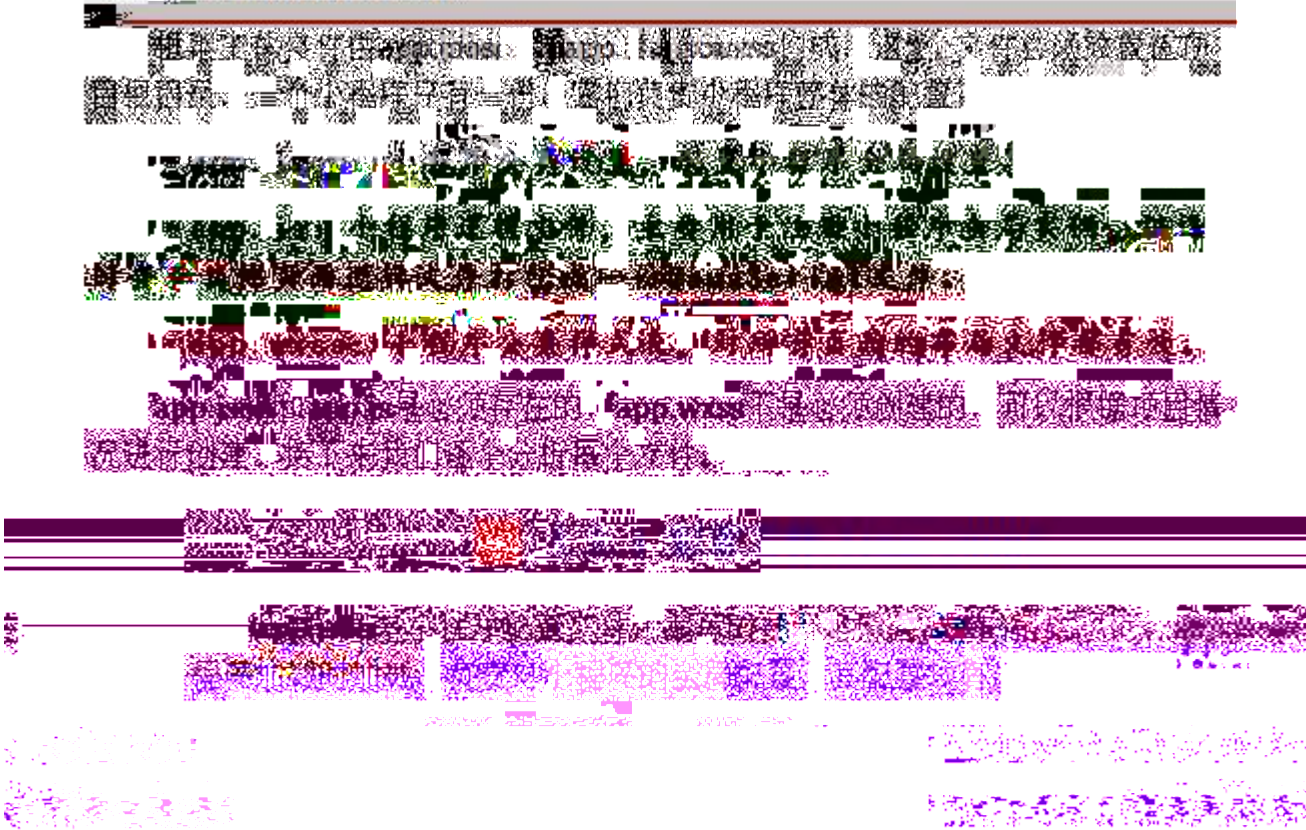
和整体样式，小程序启动时只会执行一次。这3个文件必须放在项目根目录，且文件名必须是app，其中app.json和app.js是必须的。

- 一个小程序有一个或多个页面，一个页面



第5课 框架主体文件

2.3 拆型文件示例



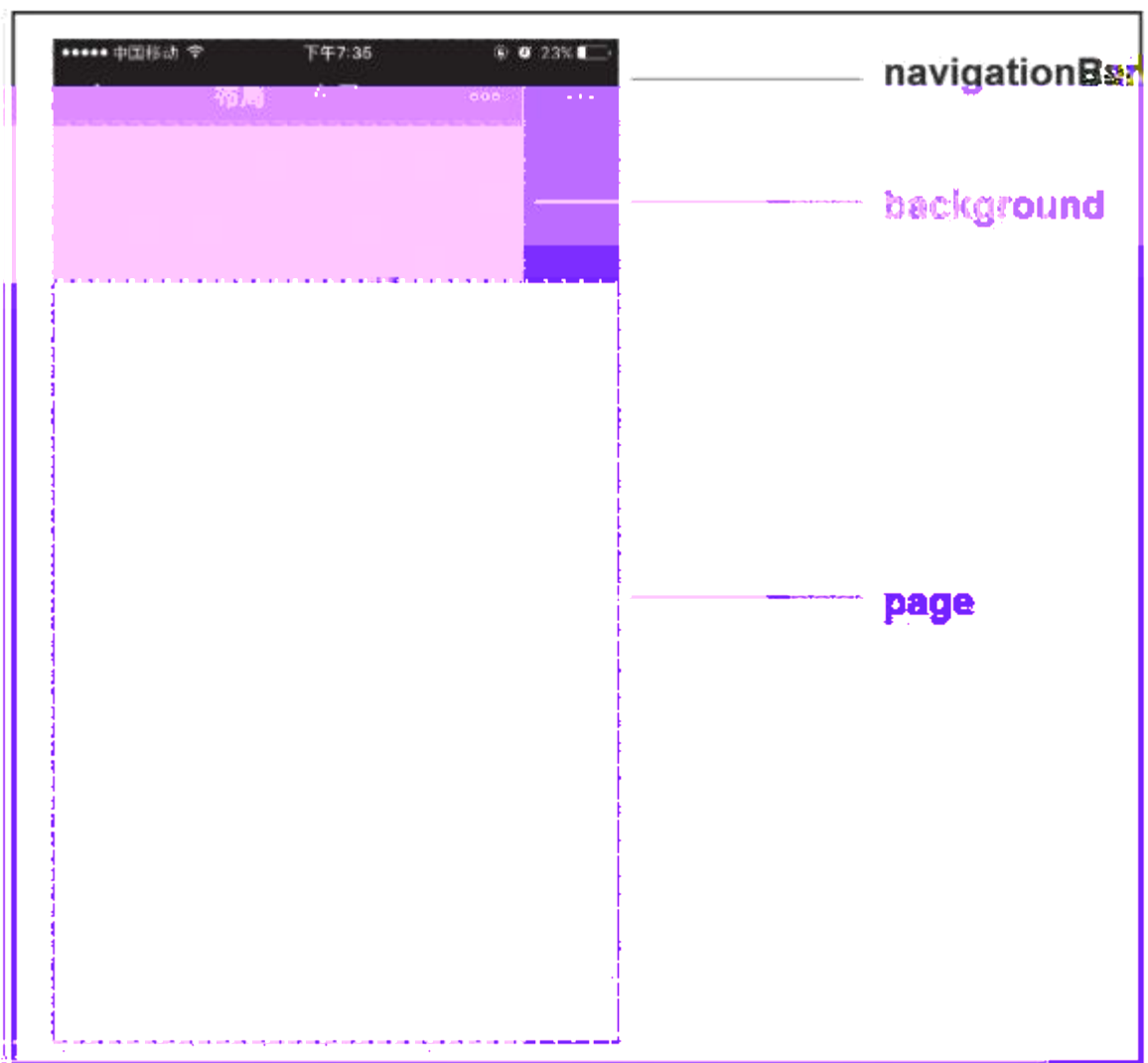
- `tabBar`: 设置tab的表现。
- `networkTimeout`: 设置网络超时时间。
- `debug`: 设置是否开启debug模式，默认关闭。

`app.json`文件内容整体结构如下：

```
{
  // 页面路径设置
  "pages": [],
  // 默认页面的窗口设置
  "window": {},
  // 底部tab设置
  "tabBar": {},
  // 设置网络请求API的超时时间
  "networkTimeout": {},
  // 是否为debug模式
  "debug": false
}
```

1. pages配置

`pages`负责注册小程序页面，必须填写，`value`值为一个包含页面路径的数



HexColor (十六进制颜色值), 必填项。

• backgroundColor: tab的背景色



• selectedColor: 选中tab的背景色



• selectedTextColor: 选中tab的字体颜色



• selectedTextSize: 选中tab的字体大小



```
"backgroundColor": "#ffffff",
```

```
"borderStyle": "black",
```

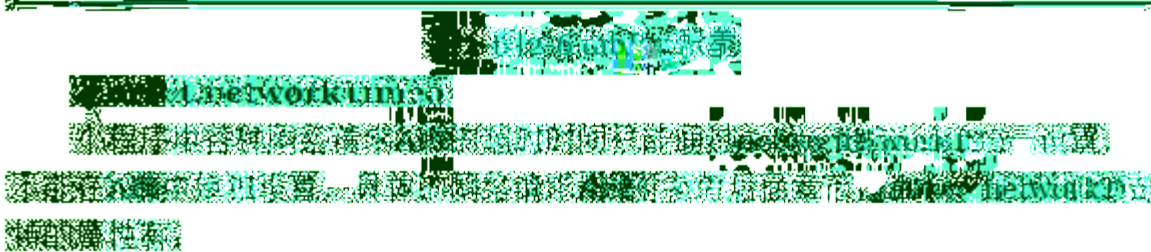
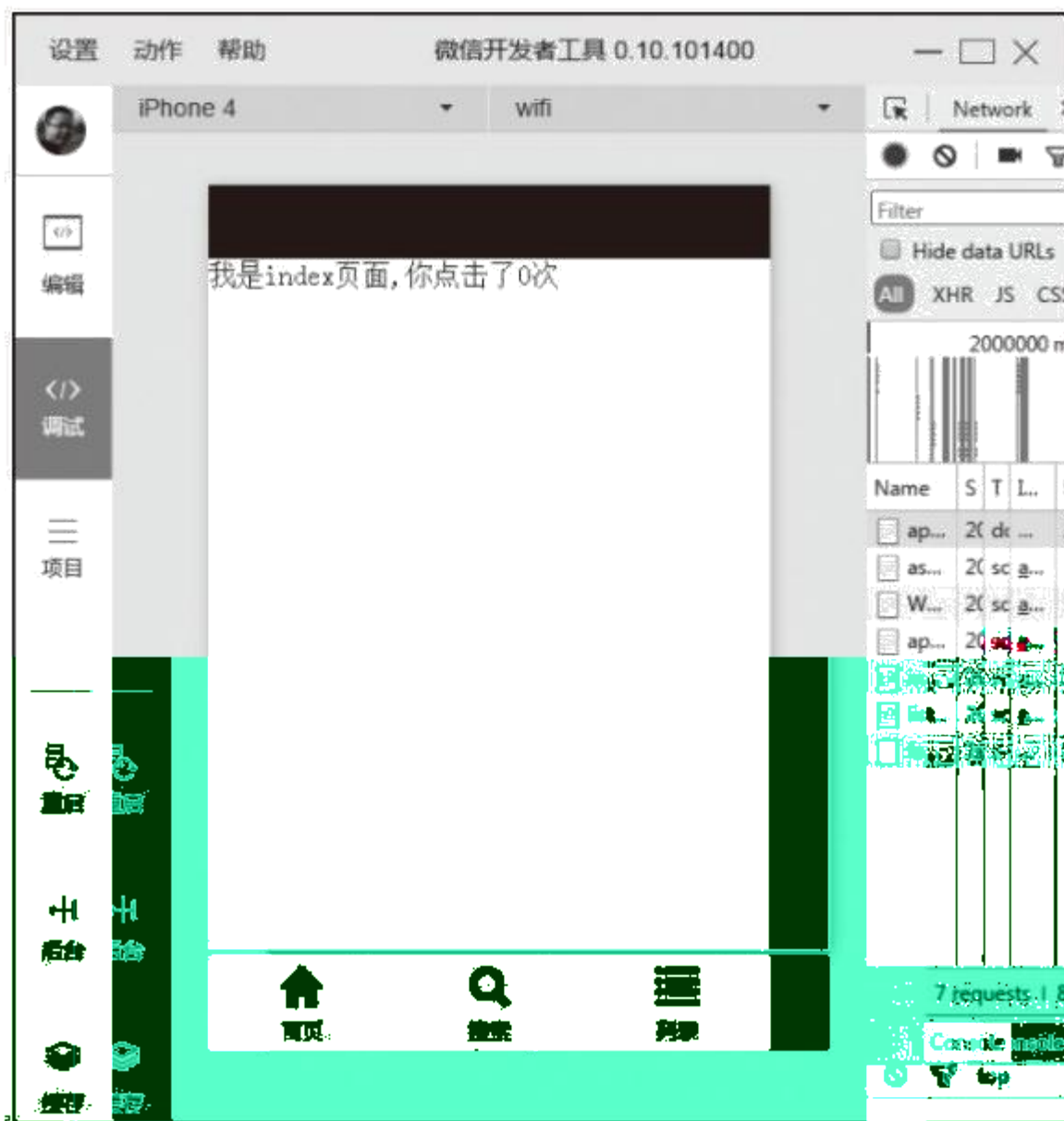
```
"list": [
```

```
{
```

```
  "iconPath": "images/home.png",
```



```
  }
```



• request: 设置wx.request的超时时间, 单位毫秒。

connectSocket: 设置wx.connectSocket的超时时间, 单位毫秒。

getAppVersion: 获取小程序版本号。

getSystemInfo: 获取系统信息, 包括系统名称、版本、品牌、型号、语言、设备像素比、屏幕宽度、屏幕高度、全屏模式、是否刘海屏、机型。

getSystemInfoSync: 同步获取系统信息。

getDeviceInfo: 获取设备信息。

getDeviceInfoSync: 同步获取设备信息。

getDeviceModel: 获取设备型号。

getDeviceModelSync: 同步获取设备型号。

getDevicePixelRatio: 获取设备像素比。

getDevicePixelRatioSync: 同步获取设备像素比。

getDeviceScreenSize: 获取设备屏幕尺寸。

getDeviceScreenSizeSync: 同步获取设备屏幕尺寸。

getDeviceFullSize: 获取设备全屏尺寸。

getDeviceFullSizeSync: 同步获取设备全屏尺寸。

getDeviceFullSizeAndOrientation: 获取设备全屏尺寸和方向。

getDeviceFullSizeAndOrientationSync: 同步获取设备全屏尺寸和方向。

getDeviceFullSizeAndOrientationWithScreenHeight: 获取设备全屏尺寸、方向及屏幕高度。

getDeviceFullSizeAndOrientationWithScreenHeightSync: 同步获取设备全屏尺寸、方向及屏幕高度。

getDeviceFullSizeAndOrientationWithScreenHeightAndType: 获取设备全屏尺寸、方向、屏幕高度及屏幕类型。

getDeviceFullSizeAndOrientationWithScreenHeightAndTypeSync: 同步获取设备全屏尺寸、方向、屏幕高度及屏幕类型。

getDeviceFullSizeAndOrientationWithScreenHeightAndTypeAndLanguage: 获取设备全屏尺寸、方向、屏幕高度、屏幕类型及语言。

getDeviceFullSizeAndOrientationWithScreenHeightAndTypeAndLanguageSync: 同步获取设备全屏尺寸、方向、屏幕高度、屏幕类型及语言。

getDeviceFullSizeAndOrientationWithScreenHeightAndTypeAndLanguageAndModel: 获取设备全屏尺寸、方向、屏幕高度、屏幕类型、语言及型号。

getDeviceFullSizeAndOrientationWithScreenHeightAndTypeAndLanguageAndModelSync: 同步获取设备全屏尺寸、方向、屏幕高度、屏幕类型、语言及型号。

getDeviceFullSizeAndOrientationWithScreenHeightAndTypeAndLanguageAndModelAndBrand: 获取设备全屏尺寸、方向、屏幕高度、屏幕类型、语言、型号及品牌。

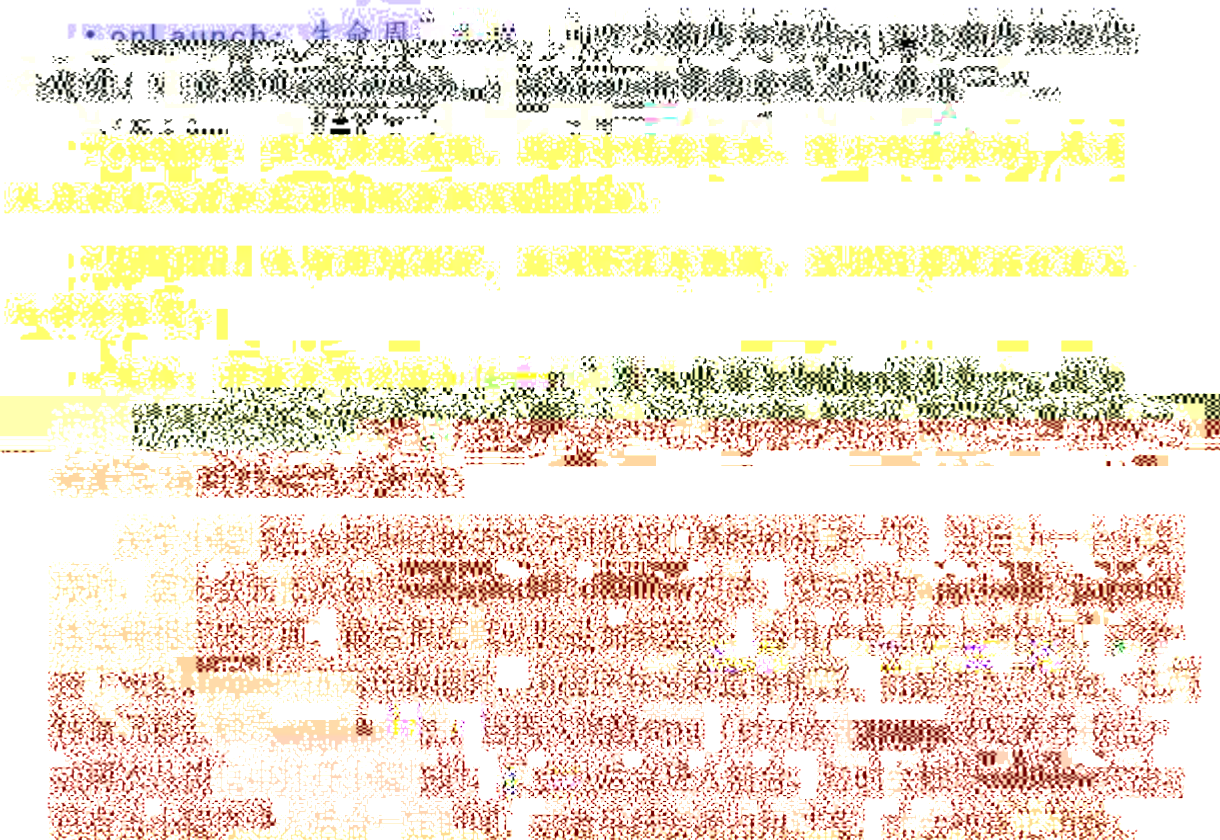
第6课 小程序逻辑文件

小程序中逻辑文件分为页面逻辑文件和小程序逻辑文件，`app.js`便是小程序逻辑文件。在这个文件中，我们可以通过`App()`函数注册小程序的全局方法和全局属性。在注册的小程序逻辑文件中，我们也可以通过`getApp()`函数

`getApp()`获取。

1. 注册小程序

`App()`函数用于注册一个小程序，参数为一个Object对象，在这个参数对象中我们可以注册自定义方法和属性供全局使用，就像在quick start项目中，我们利用`App()`注册了用户登录信息。`App()`函数必须在`app.js`中注册，且不能注册多个，其参数属性如下：



2.3.3 全局样式 (app.wxss)

app.wxss是全局样式表，对项目中每个页面都有效，可将一些系统级别的

第7课 框

2.4 框架页面文件

小程序中一个框架页面包含4个文件，同一框架页面的这4个文件必须具有相同的路径与文件名，进入小程序时或页面跳转时，小程序会根据app.json配置的路径找到对应的资源进行渲染。

- .js文件：页面逻辑文件，必要项。
- .wxml文件：页面结构文件，必要项。
- .wxss文件：页面样式文件。
- .json文件：页面配置文件。

与框架主体文件相比框架页面文件多了一种页面结构文件，其余3个文件和框架主体文件的功能类同，下面我们一一讲解每个文件作用。

2.4.1 页面配置文件

我们首先在代码编辑器中，新建一个名为“框架页面”的文件夹。

与框架主体文件不同的是，页面配置文件是非必要存在的，同时页面配置文件的配置项只有window，控制当前页面的窗口属性。

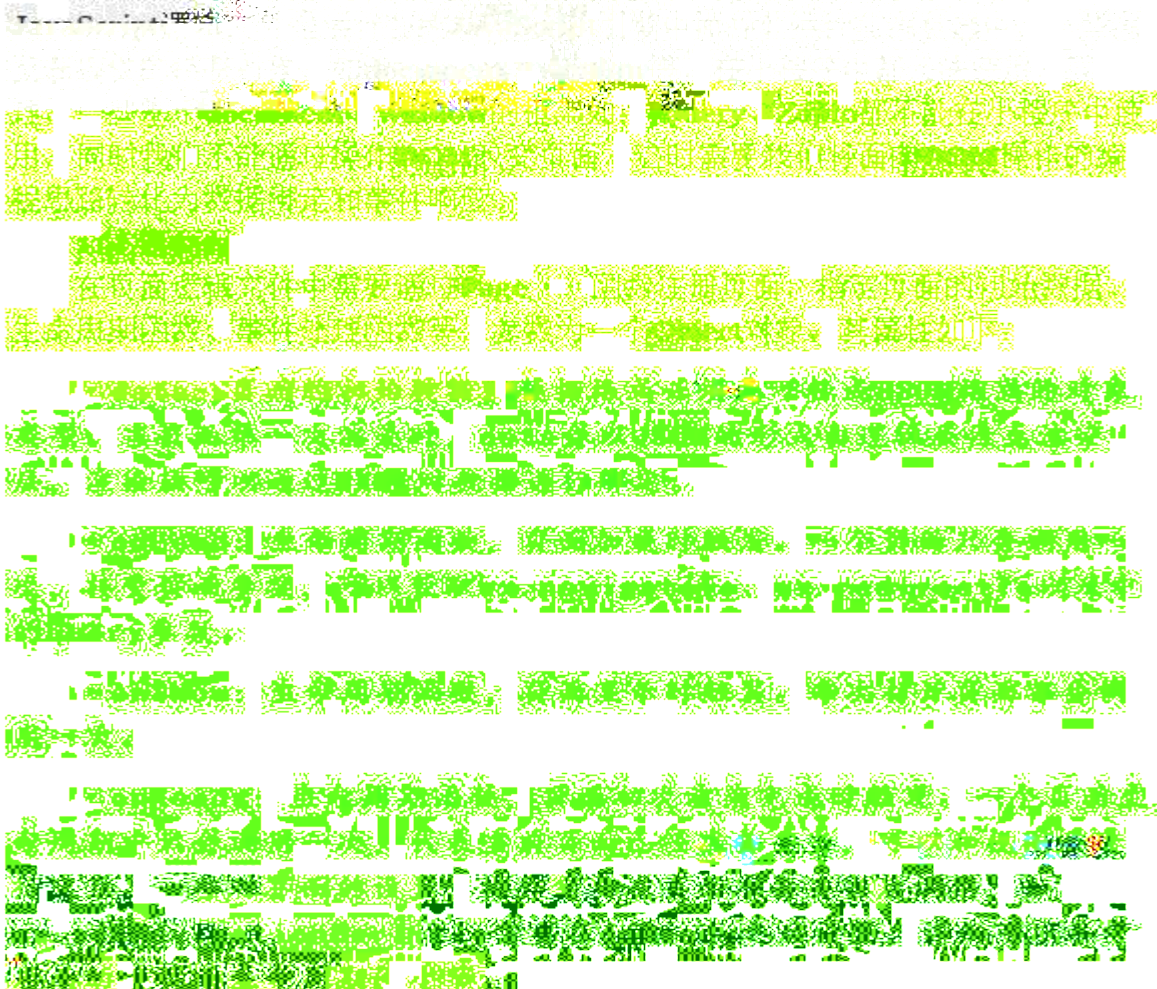
```
{  
  "navigationBarBackgroundColor": "#000000",  
  "navigationBarTextStyle": "black",  
  "navigationBarTitleText": "框架页面"  
}
```



2.4.2 页面逻辑文件 (JavaScript)

页面逻辑文件主要功能有：设置初始化数据，注册当前页面生命周期函数，注册事件处理函数等。小程序的逻辑层文件是JavaScript文件，所有的逻辑文件，包括app.js，最终将会打包成一个js文件，在小程序启动时运行，直到小程序销毁，类似于ServiceWorker，所以逻辑层也称为App Service。

在小程序中，每个逻辑文件有独立的作用域，并具备模块化能力。由于



• onHide: 生命周期函数，页面隐藏时触发。

• onUnload: 生命周期函数，页面卸载时触发。

• onReady: 生命周期函数，页面初次渲染完成时触发。

发。使用时需要将app.json配置中window的enablePullDownRefresh

属性设置为true，即可开启下拉刷新功能。

在微信小程序中，下拉刷新功能是通过调用wx.refreshPage

接口实现的。该接口用于刷新当前页面，使其重新从服务器

获取数据。调用该接口时，需要提供刷新成功的回调函数。

以下是一个使用wx.refreshPage接口的示例代码：

```
wx.refreshPage({  
  url: '/pages/index/index',  
  success: function() {  
    // 刷新成功后的回调函数  
  }  
})
```

在以上代码中，url属性指定了要刷新的页面路径，success

属性指定了刷新成功的回调函数。

需要注意的是，调用wx.refreshPage接口时，必须在主程

序中调用，不能在子进程中调用。

此外，调用该接口时，还需要注意以下几点：

1. 调用该接口时，必须在主进程中调用，不能在子进程中

调用。

2. 调用该接口时，需要提供刷新成功的回调函数。

3. 调用该接口时，需要提供要刷新的页面路径。

4. 调用该接口时，需要提供刷新成功的回调函数。

5. 调用该接口时，需要提供刷新成功的回调函数。

6. 调用该接口时，需要提供刷新成功的回调函数。

7. 调用该接口时，需要提供刷新成功的回调函数。

8. 调用该接口时，需要提供刷新成功的回调函数。

9. 调用该接口时，需要提供刷新成功的回调函数。

10. 调用该接口时，需要提供刷新成功的回调函数。

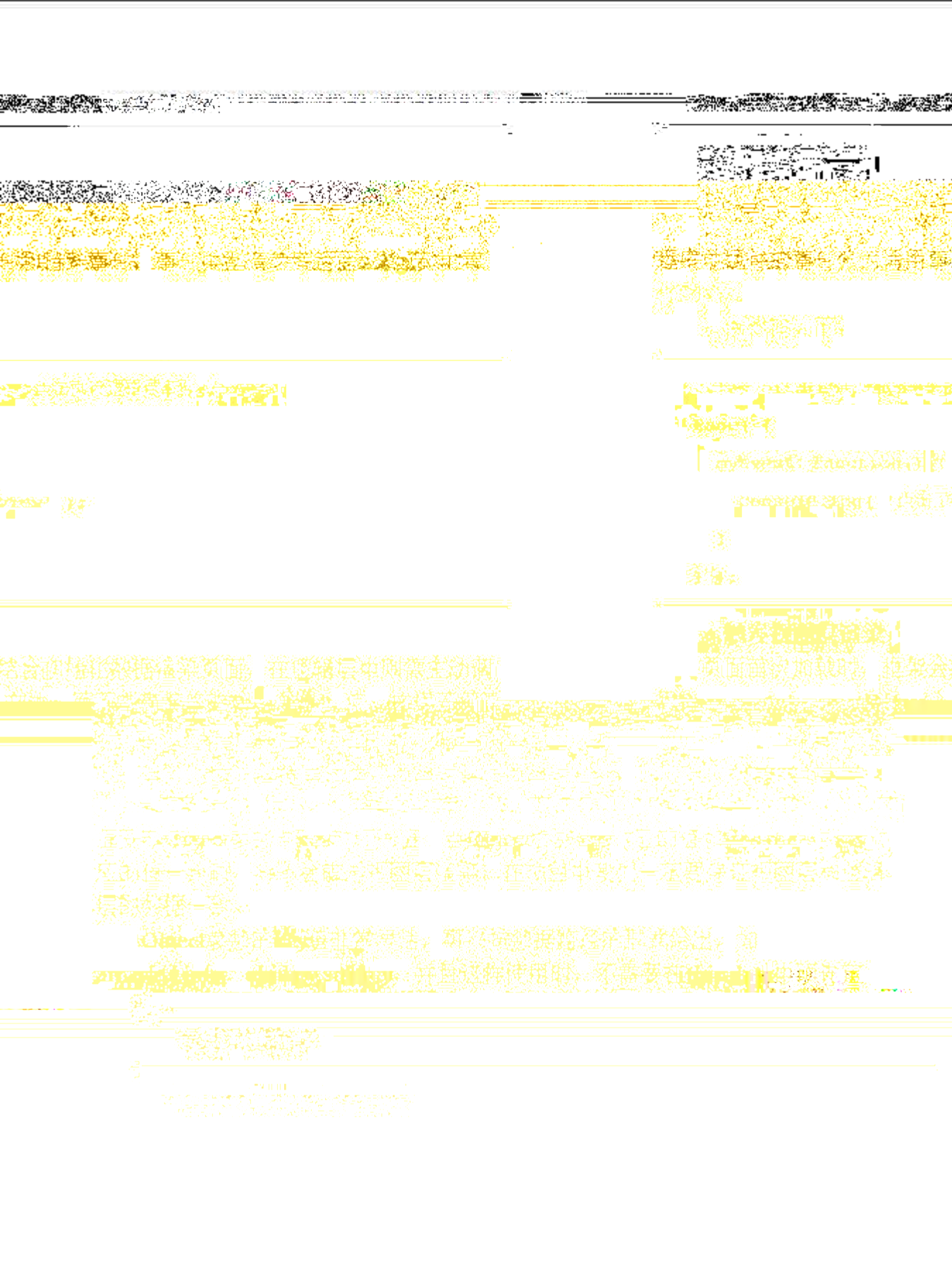
11. 调用该接口时，需要提供刷新成功的回调函数。

12. 调用该接口时，需要提供刷新成功的回调函数。

13. 调用该接口时，需要提供刷新成功的回调函数。

14. 调用该接口时，需要提供刷新成功的回调函数。

15. 调用该接口时，需要提供刷新成功的回调函数。



```
<button bindtap="changeText">修改普通数据</button>
```

```
<view>{{object.subObject.objectText}}</view>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
</page>
```

```
        'object.subObject.objectText': 'new object data'
    });
},
changeArrayText: function() {
    this.appData = {
        'arrayText': 'new array text'
    };
    console.log('changeArrayText');
    console.log(this.appData);
}
};
```

Object 的继承

Object 的继承有很多种方法，本文主要介绍原型链继承和类继承。原型链继承是 JavaScript 中最常用的继承方式，而类继承则是 ES6 引入的。本文还介绍了如何避免原型链继承的缺点，以及如何使用类继承。

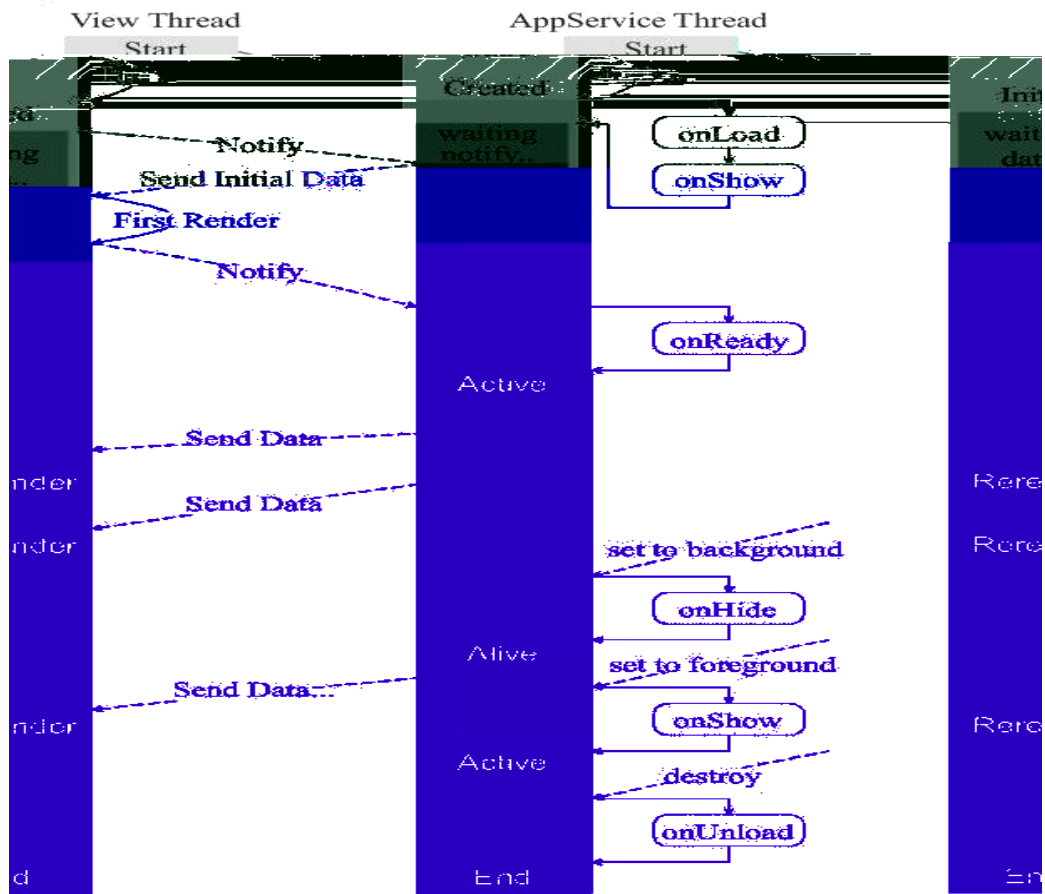
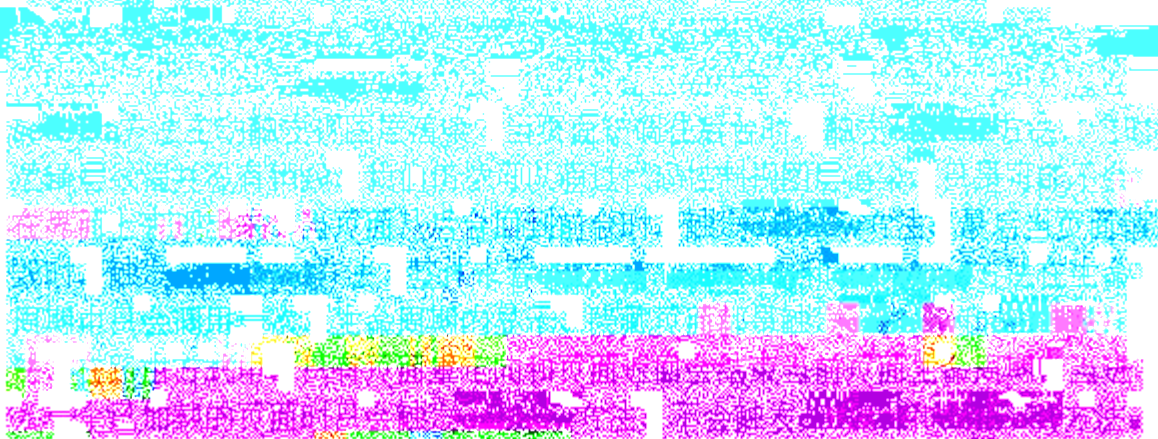


图2-8 Page实例的生命周期

如图2-8所示，线程启动后视图层和逻辑层都



第9课 页面结构文件

2.4.3 页面结构文件 (WXML)

WXML (WeiXin Markup Language) 是框架设计的一套标记语言，用于渲染界面，WXML的渲染原理和React Native思路一致，通过

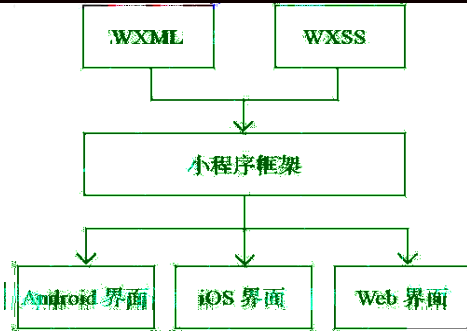


图2-9 界面渲染示意图

从图中我们能看出，WXML语言最终会转译为宿主端对应的语言，所以

WXML中所使用的标签一定是小程序

直接作为字符串输出使用，可作用于内容、组件属性。

在中文输入时，会有延迟 delay 的反馈，这是由输入法造成的。

图 1-1-10

图 1-1-11

图 1-1-12

图 1-1-13

图 1-1-14

图 1-1-15

图 1-1-16

图 1-1-17

图 1-1-18

图 1-1-19

图 1-1-20

图 1-1-21

图 1-1-22

图 1-1-23



```
<view>{{ showContent ? '显示文本' : '不显示文本'}}</view>
```

```
<!-- 算数运算符 -->
```

```
<view>{{ num1 + num2 }} + 1 + {{ num3 }} = ? </view>
```

```
<!-- 字符串运算 -->
```

```
<view>{{ "name:" + name }}</view>
```

逻辑判断

字符串拼接和逻辑判断



不显示文本
3 + 1 + 3 = 2
name : weixin
true
12.12.2

不显示文本
3 + 1 + 3 = 2
name : weixin
true
12.12.2

不显示文本
3 + 1 + 3 = 2
name : weixin
true
12.12.2

不显示文本
3 + 1 + 3 = 2
name : weixin
true
12.12.2

最终页面组合成的对象为[0, 2, 3, 'stringtype']。

对象组合有3种组合方式，这里我们以数据注入模板为例。

第一种，直接通过模板：

```

<!-- 模板 -->
<table border="1">
  <tr>
    <td>{{myvar}}</td>
  </tr>
</table>

```

第二种，通过“<include>”指令，在模板中引入其他模板：

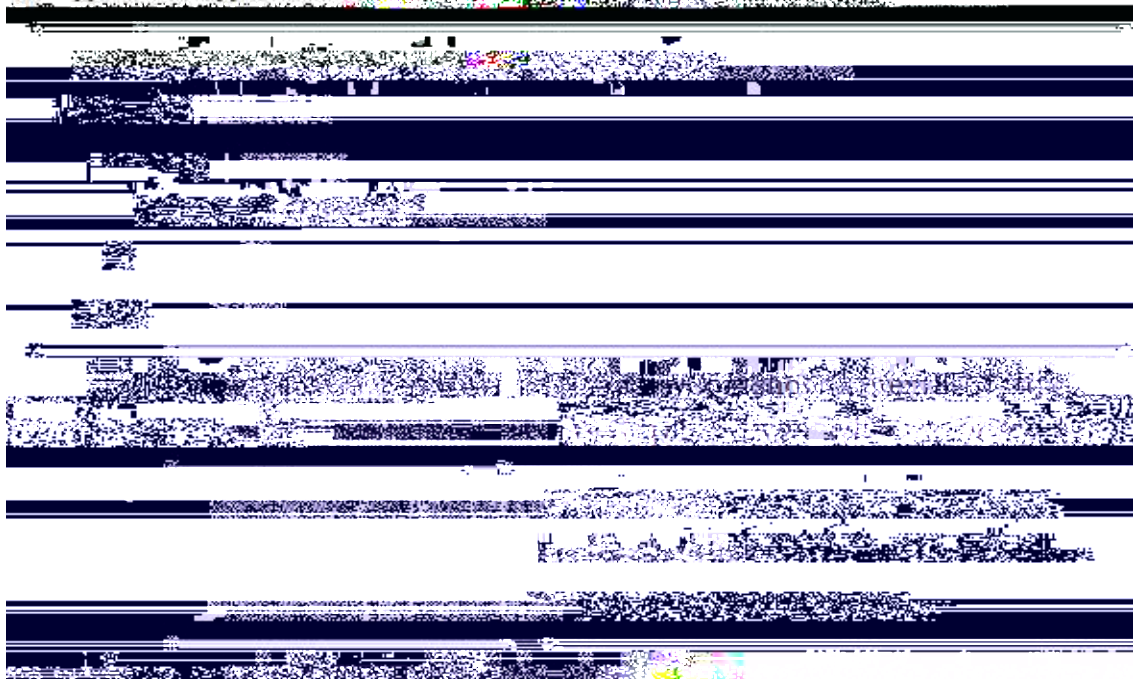
第三种，通过“<include>”指令，在模板中引入其他模板中：

```

<!-- 模板 -->
<table border="1">
  <tr>
    <td>{{myvar}}</td>
  </tr>
</table>

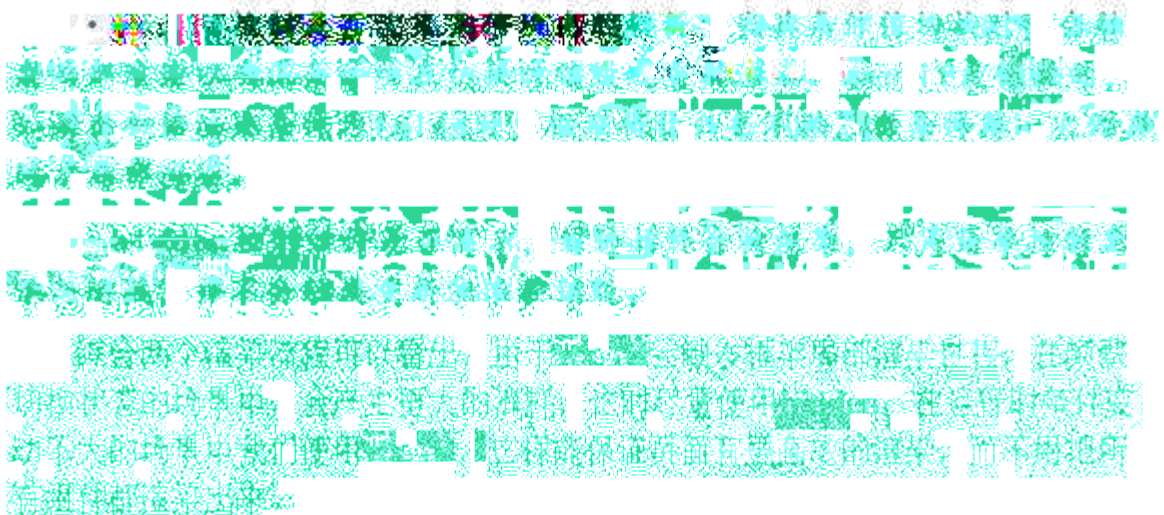
```


除了简单的数据绑定，我们常常会使用逻辑分支，这时候可以使用wx:if="{{判断条件}}"来进行条件渲染，当条件成立时渲染该代码块。



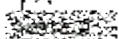
(3) wx:if与hidden

除了wx:if组件，也可以通过hidden属性控制组件是否显示。对于这两种方式的疑问，这两种方式该怎样取舍，这里我们整理了两种方式的区别：



第11课 列表渲染

组件的 `wx.foreach` 控制 [微信小程序官方文档](#) [董嘉远](#) [微信小程序官方文档](#)



```
item="myItem">
```

```
  {{myIndex}}: {{myItem.name}}
```

2025年 10月

2025年 10月



图2-12 列表渲染示例1

渲染结果如图2-13所示。

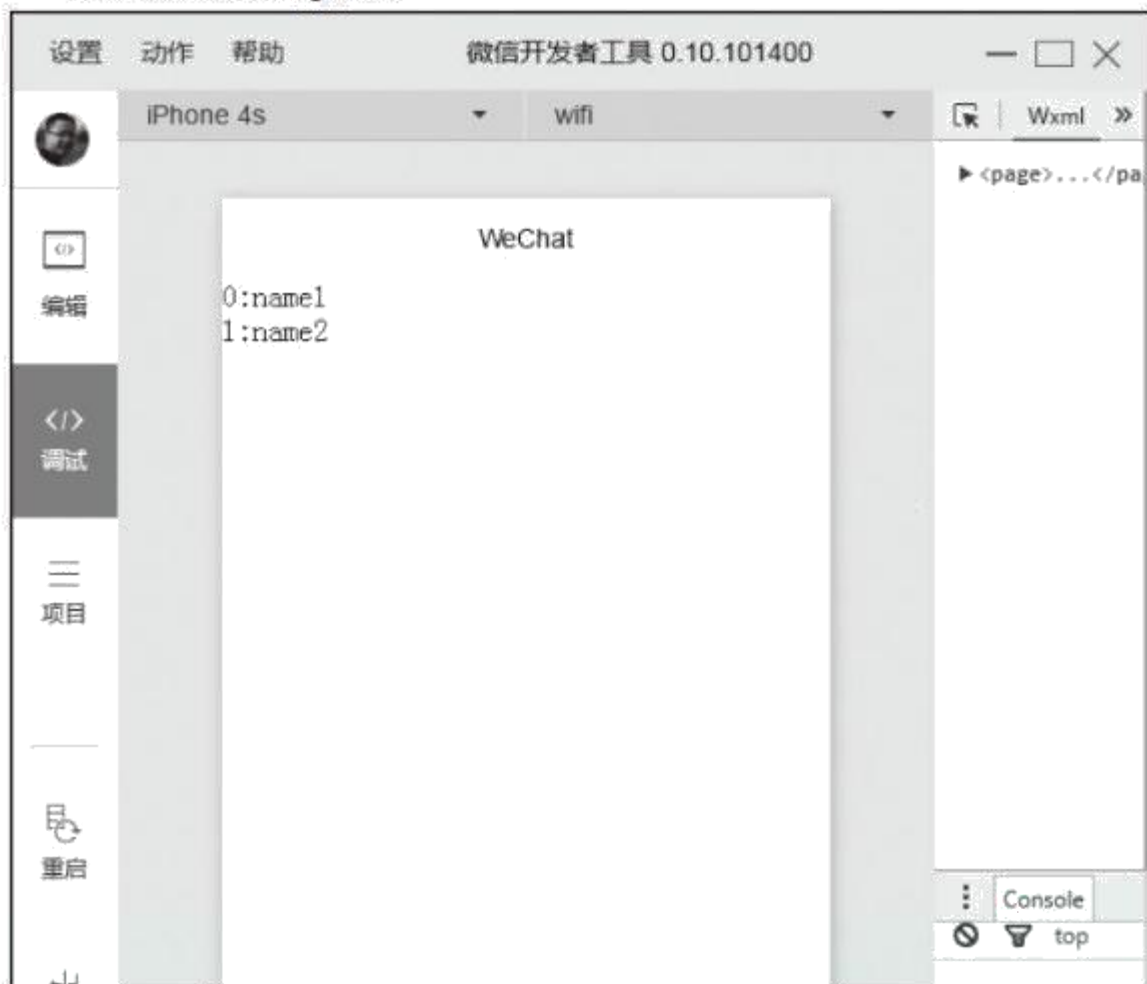


图2-13 列表渲染示例2

普通渲染由我们自己去修改index.html 列表变量名，当然，对于嵌套使用叶

子节点更复杂，我们可以在这里看看，目前我们暂时不讨论。

```
<view wx:for="{{array}}" wx:for-item="{{item}}" wx:for-index="{{index}}">
  <text class="list-item">{{item}}</text>
</view>
```

图2-14

```
</block>
</view>
Page({
  data: {
    myArray: [
      [1, 2, 3],
      [4, 5, 6],
      [7, 8, 9]
    ]
  }
});
```

渲染效果如图2-14所示。

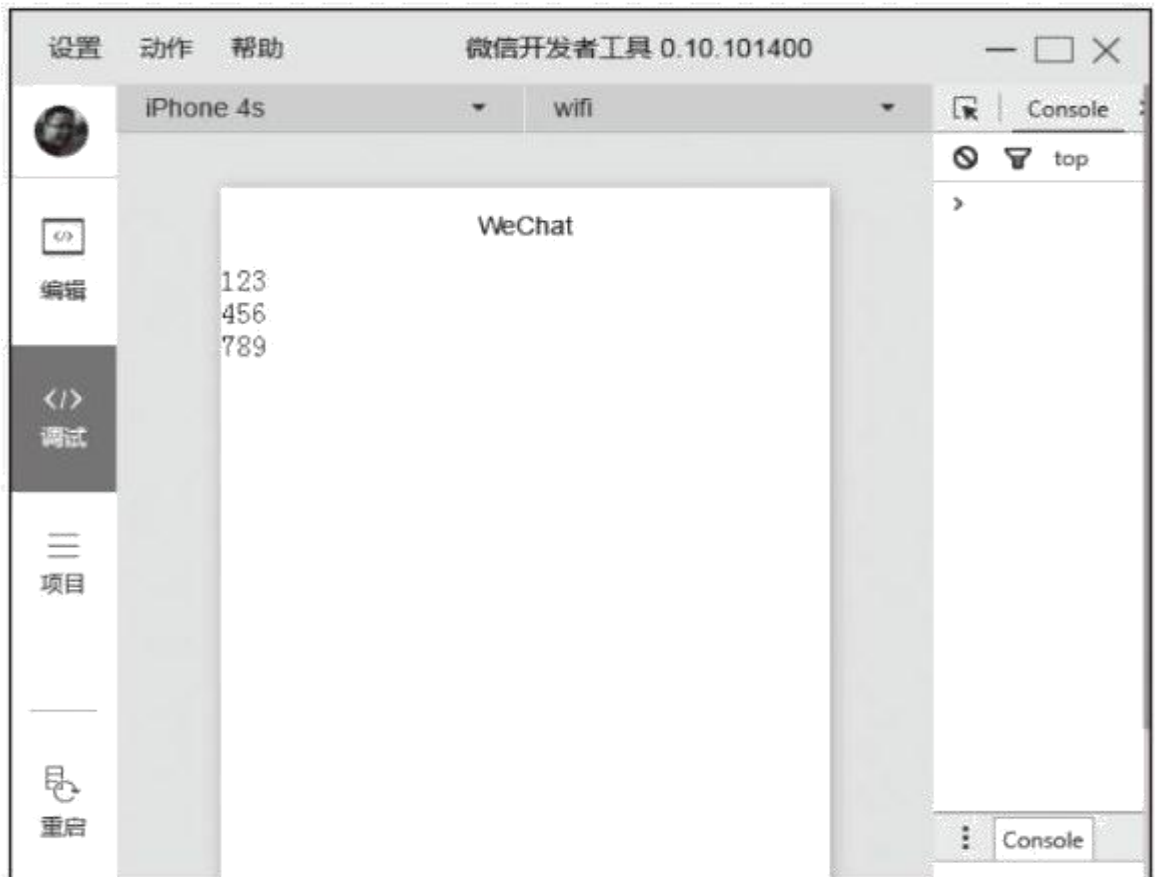


图2-14 列表渲染示例3

在本示例中，我们使用了`<block/>`标签，和`block wx: if`一样，`wx: for`可以直接在`<block/>`标签上使用，以渲染一个包含多个节点的列表。

第12课 模板

在项目过程中，常常会遇到某些相同的结构在不同的地方反复出现，这时可以创建模板，将公共代码放置到一个模板中，在需要的时候直接引用模板，避免重复开发，提高开发效率。

1. 创建模板

在开发模板的过程中，可以在模板的根节点添加任意数量的子节点，如 `<view></view>` 的 `view` 属性，指定模板，如代码清单 12-1 所示。

```
<template>  
  <view>内容</view>  
  <view>{{content}}</view>  
</template>
```

(2) 使用模板



模板的根节点为 `template`，如代码清单 12-1 所示。

模板的根节点包含子节点。

模板的根节点包含子节点。

模板的根节点包含子节点。

模板的根节点包含子节点。

模板的根节点包含子节点。

模板的根节点包含子节点。

```
<template id="myTemplate" data="/content/内容" name="myName">
```

```
</template>
```

```
</script>
```

```
<script type="text/javascript">  
    $(function() {  
        <!-- 内容 -->  
    });  
</script>
```

```
</body>  
</html>
```

```
</pre>
```

```
</code>
```

```
</pre>
```

```
</code>
```

执行效果如图2-15所示。

模板的默认名称为

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```

```
<code>myTemplate</code>
```



图2-15 模板示例

渲染结果如图2-16所示。

iPhone 4s

wifi

Wxml

> <page>...</pa

WeChat

a template content

template content



调试



项目



微信

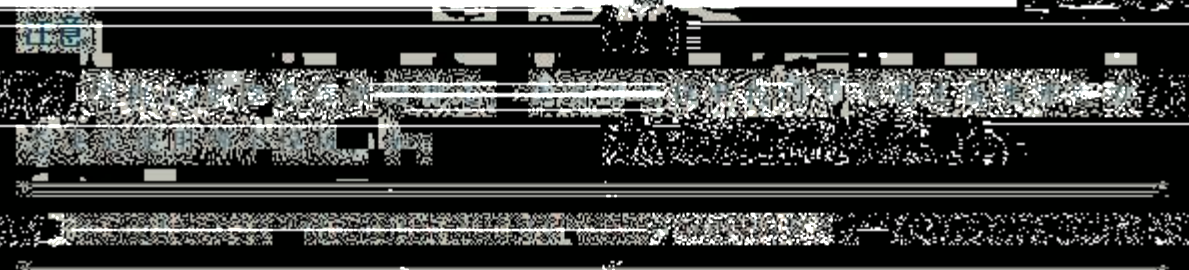


微信

Canvas



图标的拖拽使用模板



第13课 事件

WXML事件绑定是WXML中非常重要的一个特性。它允许我们动态地绑定事件到WXML元素上，从而实现交互功能。官方对WXML事件的定义如下：

- 事件是视图层接收的交互行为。事件绑定是WXML中最重要的特性之一，它允许我们动态地绑定事件到WXML元素上，从而实现交互功能。

在WXML中，事件绑定是通过在元素属性中使用on:前缀来实现的。例如，我们可以为元素绑定click事件，当用户点击该元素时，就会触发绑定的事件处理函数。

事件绑定的基本语法如下：

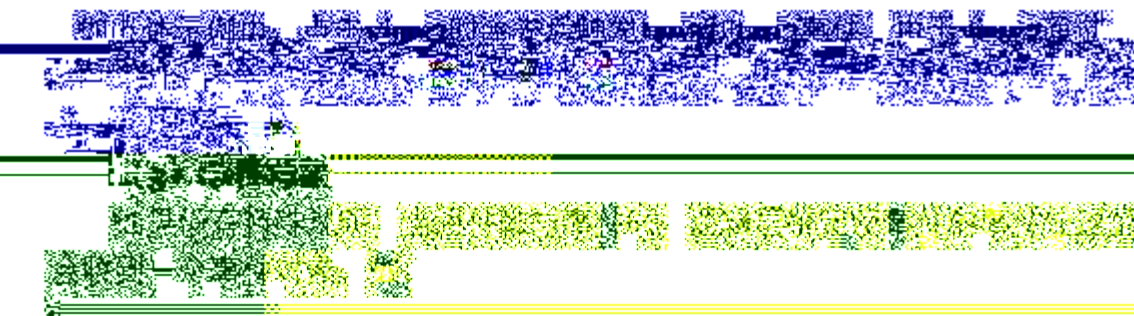
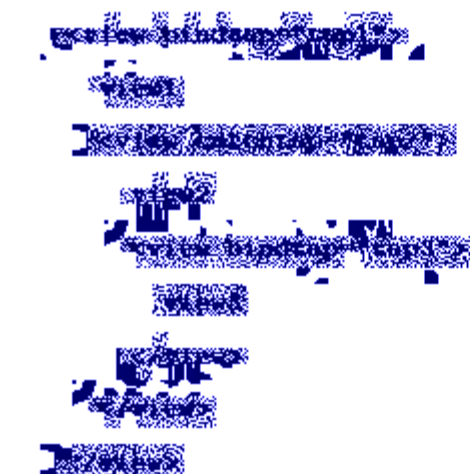
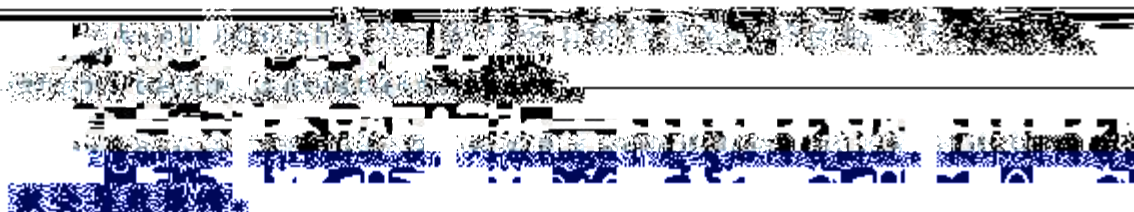
```
<view class="container">
  <button class="button" type="button" on:click="handleClick">
    点击我
  </button>
</view>
```

在上述代码中，on:click表示绑定click事件，handleClick表示事件处理函数的名称。当用户点击button元素时，就会调用handleClick函数。

除了click事件，WXML还支持许多其他类型的事件，如tap、touchstart、touchend等。我们可以在WXML元素上使用on:前缀来绑定这些事件。

事件绑定是WXML中实现交互功能的关键。通过合理地使用事件绑定，我们可以创建出具有丰富交互性的WXML视图。

在之前内容中，已经多次实现事件绑定，大家应该比较熟悉了。事件绑定的



myevangelical.com

THE
EVANGELICAL

WORLD

WORLD

THE
EVANGELICAL
WORLD

THE
EVANGELICAL
WORLD

THE
EVANGELICAL
WORLD

THE
EVANGELICAL
WORLD

THE
EVANGELICAL
WORLD

THE
EVANGELICAL
WORLD

THE
EVANGELICAL
WORLD

THE
EVANGELICAL
WORLD

THE
EVANGELICAL
WORLD

THE
EVANGELICAL
WORLD

THE
EVANGELICAL
WORLD

THE
EVANGELICAL
WORLD

THE
EVANGELICAL
WORLD

THE
EVANGELICAL
WORLD

THE
EVANGELICAL
WORLD

THE
EVANGELICAL
WORLD

```
"touches": [
```

```
{
```

```
  "identifiant": 0,
```

```
  "pageX": 15,
```



```
},
```



```
],
```

```
{
```



```
},
```



```
],
```

```
],
```



```
],
```

```
],
```



- `id`: 事件源组件的 `id`。

- `tagName`: 事件源组件的类型。

- `dataset`: 事件源组件上由 `data-` 开头的自定义属性组成的对象。

```
import { createEventSource } from '@vueuse/core'
```

```
const source = createEventSource('/api')
```

```
source.on('message', (message) => {
```

```
  console.log(message)
```

```
})
```

```
source.close()
```

```
const source = createEventSource('/api')
```

```
source.on('message', (message) => {
```

```
  console.log(message)
```

```
})
```

```
source.close()
```

```
source.on('message', (message) => {
```

```
  console.log(message)
```

```
})
```

```
source.close()
```

```
source.on('message', (message) => {
```

```
  console.log(message)
```

```
})
```

```
source.close()
```

```
source.on('message', (message) => {
```

```
  console.log(message)
```

```
})
```

```
source.close()
```

```
source.on('message', (message) => {
```

```
  console.log(message)
```

```
})
```

```
source.close()
```

```
source.on('message', (message) => {
```

```
  console.log(message)
```

```
})
```

```
source.close()
```

CustomEvent为自定义事件对象（继承BaseEvent），只有一个属性

`CustomEvent.prototype = new BaseEvent();`

```
function CustomEvent(eventType, data) {
    this.type = eventType;
    this.data = data;
}
CustomEvent.prototype = new BaseEvent();
```

```
function BaseEvent() {
    this.type = null;
    this.data = null;
}
```

```
function CustomEvent(eventType, data) {
    this.type = eventType;
    this.data = data;
}
CustomEvent.prototype = new BaseEvent();
```

```
function BaseEvent() {
    this.type = null;
    this.data = null;
}
```

```
function CustomEvent(eventType, data) {
    this.type = eventType;
    this.data = data;
}
CustomEvent.prototype = new BaseEvent();
```

```
function BaseEvent() {
    this.type = null;
    this.data = null;
}
```

```
function CustomEvent(eventType, data) {
    this.type = eventType;
    this.data = data;
}
CustomEvent.prototype = new BaseEvent();
```

```
function BaseEvent() {
    this.type = null;
    this.data = null;
}
```

第14课 引用

一个WXML可以通过import或include引入其他WXML文件，两种方式都能引入WXML文件，区别在于import引入WXML文件后只接受模板的定义，而在include引入WXML文件时，就使用引入程序文件中的模板。include则是引入文件中除<template/>以外的代码直接拷贝到<include/>位置。整体来说

import引入模板文件

```
<import src="a.xml" />
```

include引入模板文件

```
<include src="a.xml" />
```

include引入代码文件

```
<include src="a.xml" type="text" />
```

```
<include src="a.xml" type="text" />
```

```
<include src="a.xml" type="text" />
```

```
<include src="a.xml" type="text" />
```

```
<include src="a.xml" type="text" />
```

```
<include src="a.xml" type="text" />
```

```
<include src="a.xml" type="text" />
```

```
<include src="a.xml" type="text" />
```

```
<include src="a.xml" type="text" />
```

```
<include src="a.xml" type="text" />
```

```
<include src="a.xml" type="text" />
```

```
<include src="a.xml" type="text" />
```

```
<include src="a.xml" type="text" />
```

```
<include src="a.xml" type="text" />
```

```
<include src="a.xml" type="text" />
```

```
<include src="a.xml" type="text" />
```

```
<include src="a.xml" type="text" />
```

```
<include src="a.xml" type="text" />
```

```
<include src="a.xml" type="text" />
```

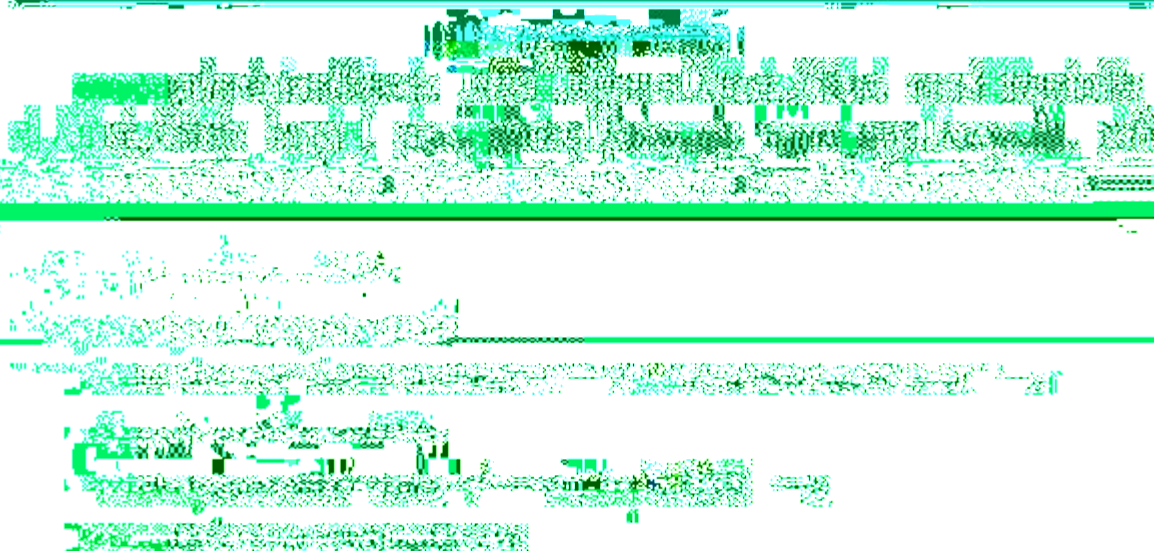
```
<include src="a.xml" type="text" />
```

```
<include src="a.xml" type="text" />
```

```
<include src="a.xml" type="text" />
```

```
<include src="a.xml" type="text" />
```

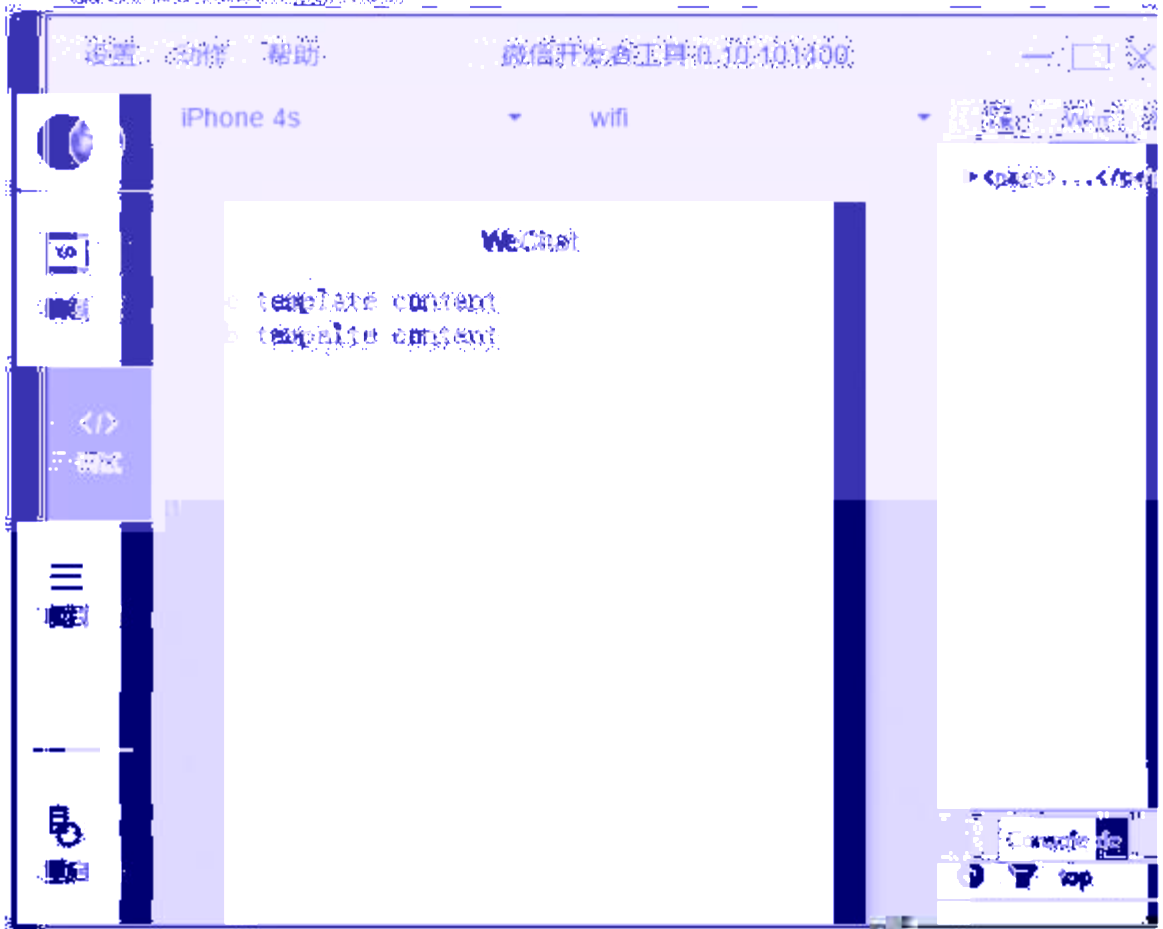
```
<include src="a.xml" type="text" />
```

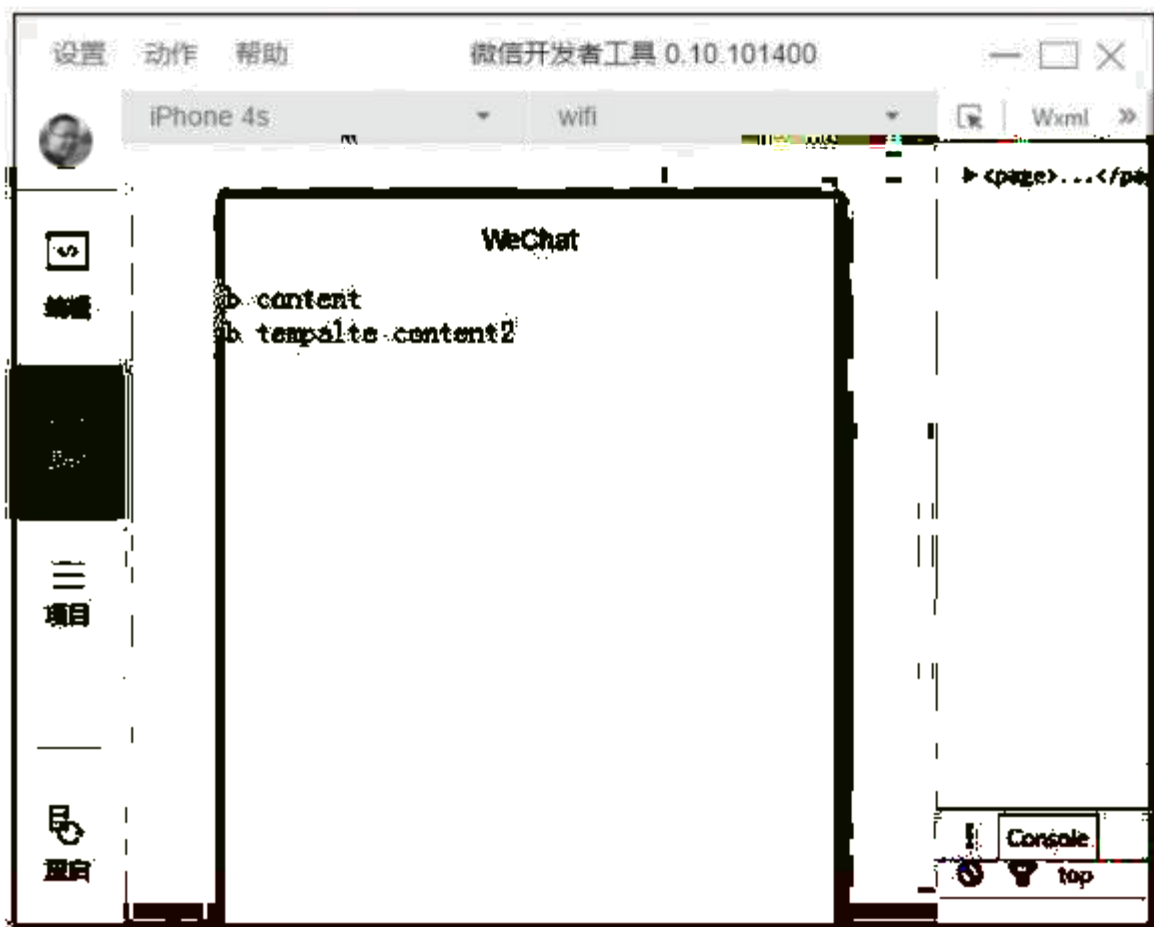


```
<template is="cTemplate"/>
<view>b tempalte content</view>
</template>
```

```
<template is="cTemplate"/> <!-- import时被忽略 -->
<template name="cTemplate">
  <view>c template content</view>
</template>
```

渲染效果如图 10-10-10 所示。





第15课 页面样式文

2.4.4 页面样式文件 (WXSS)

WXSS (WeiXin Style Sheets) 是基于CSS拓展的样式语言, 用于描述WXML的组件样式, 决定WXML的组件该怎么显示, 它具有CSS的大部分特性, 在CSS基础上WXSS拓展了尺寸单位、样式导入特性, 对CSS选择器属性上做了部分兼容。

使用CSS属性。—— 在代码中, 我们使用 `class="red"` 来给 `div` 添加一个 `red` 的类名, 在CSS中, 我们使用 `color:red` 来给 `div` 添加一个 `color` 属性, 使其文字颜色为红色。



... (faded text) ...

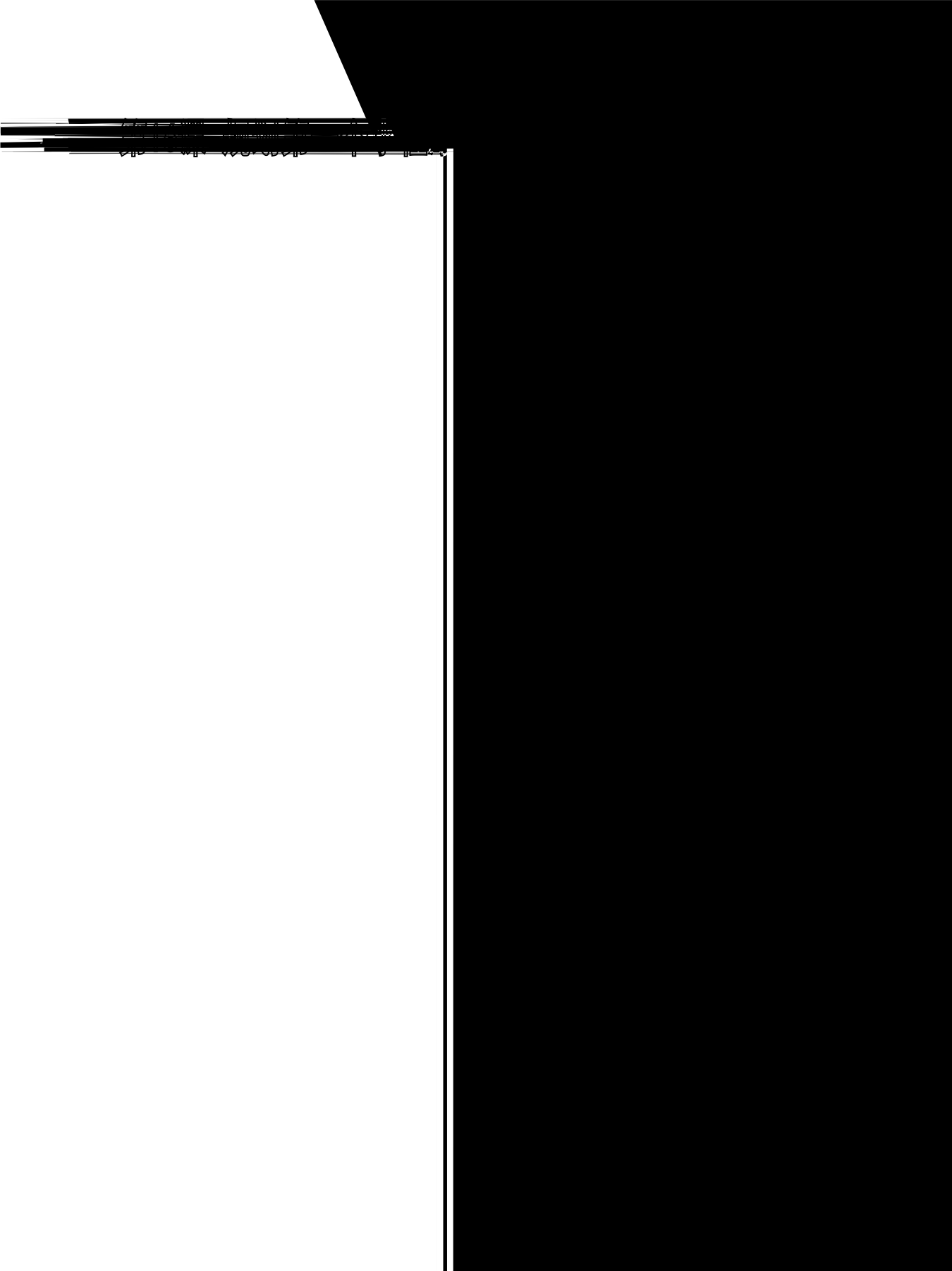
...
...
...

... (faded text) ...

... (faded text) ...

...	...
...	...
...	...

... (faded text) ...



第17课 制作小程序

第18課 完