

微信小程序之旅

尚硅谷前端微信小程序课程

版本：V 2.0

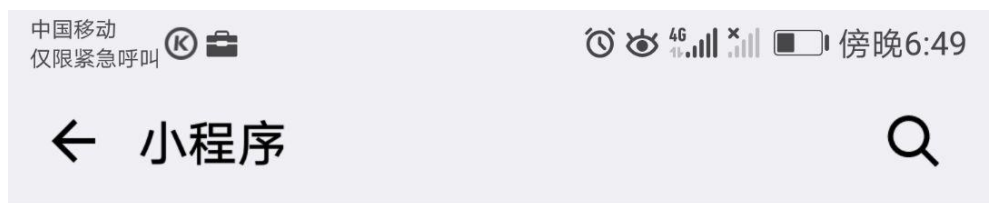
讲师：闫志勇

第 1 章 什么是小程序？

1. 2017 年度百度百科十大热词之一
2. 微信小程序，简称小程序，英文名 Mini Program，是一种不需要下载安装即可使用的应用（张小龙对其的定义是无需安装，用完即走，实际上是需要安装的，只不过小程序的体积特别小，下载速度很快，用户感觉不到下载的过程）
3. 小程序刚发布的时候要求压缩包的体积不能大于 1M，否则无法通过，在 2017 年 4 月做了改进，由原来的 1M 提升到 2M；
4. 2017 年 1 月 9 日 0 点，万众瞩目的微信第一批小程序正式低调上线。

第 2 章 小程序可以干什么？

1. 同 App 进行互补，提供同 app 类型的功能，比 app 使用方便简洁
2. 通过扫一扫或者在微信搜索即可下载
3. 用户使用频率不高，但又不得不用功能软件，目前看来小程序是首选
4. 连接线上线下
5. 开发门槛低，成本低



附近的小程序



开发版



每日优鲜便利购



你说包



包你懂我



百鱼说



好友在哪儿



第 3 章 小程序开发资料

3.1 相关资料

1) 官网: <https://mp.weixin.qq.com/>

2) 微信开发工具

3

更多 [Java](#) - [大数据](#) - [前端](#) - [python](#) 人工智能资料下载, 可访问百度: 尚硅谷官网

开发工具 IDE 文件夹中获取

3) 下载地址

<https://mp.weixin.qq.com/debug/wxadoc/dev/devtools/download.html?t=2018315>

3.2 注册小程序账号

1) 有账号



2) 无账号

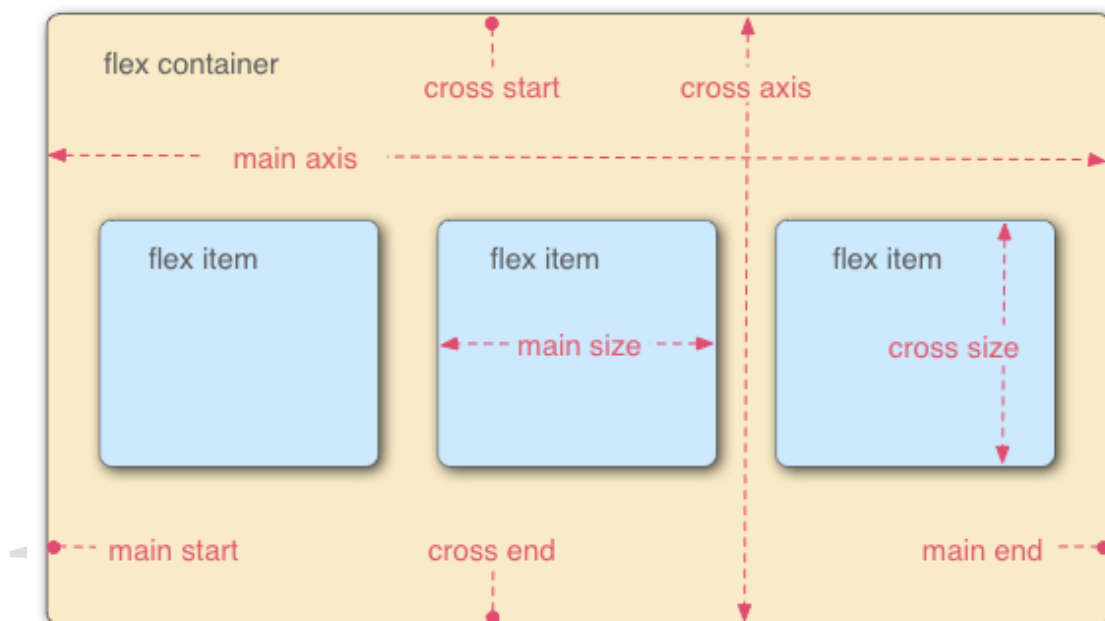


第 4 章 开发小程序储备知识

4.1 Flex 布局简介

4.1.1 什么是 flex 布局？

- 1) Flex 是 Flexible Box 的缩写，意为“弹性布局”，用来为盒状模型提供最大的灵活性。
- 2) 任何一个容器都可以指定为 Flex 布局。
- 3) display: 'flex'



4.1.2 flex 属性

- 1) flex-direction:

row（默认值）：主轴为水平方向，起点在左端。

row-reverse：主轴为水平方向，起点在右端。

column：主轴为垂直方向，起点在上沿。

column-reverse：主轴为垂直方向，起点在下沿。

4.1.3 学习地址:

<http://www.runoob.com/w3cnote/flex-grammar.html>

4.2 移动端相关知识

4.2.1 物理像素

- 1) 屏幕的分辨率
- 2) 设备能控制显示的最小单元, 可以把物理像素看成是对应的像素点

4.2.2 设备独立像素 & css 像素

设备独立像素(也叫密度无关像素), 可以认为是计算机坐标系统中的一个点, 这个点代表一个可以由程序使用并控制的**虚拟像素**(比如: CSS 像素, 只是在 android 机中 CSS 像素就不叫"CSS 像素"了而是叫"设备独立像素"), 然后由相关系统转换为物理像素。

4.2.3 dpr 比 & DPI & PPI

- 1) dpr: 设备像素比, $\text{物理像素} / \text{设备独立像素} = \text{dpr}$, 一般以 Iphon6 的 dpr 为准 $\text{dpr} = 2$
- 2) PPI: 一英寸显示屏上的像素点个数
- 3) DPI: 最早指的是打印机在单位面积上打印的墨点数, 墨点越多越清晰

	iPhone 3G/ 3GS	iPhone 4/4S	iPhone 5/5S/ 5C	iPhone 6	iPhone 6+
	3.5"	3.5"	4"	4.7"	5.5"
physic pixel	320x480	640x960	640x1136	750x1334	1080x1920 (downsampling)
logic pixel	~	~	~	~	1242x2208
logic point	320x480	320x480	320x568	375x667	414x736
scale	@1x	@2x	@2x	@2x	@3x
PPI	163	326	326	326	401
DPI	163	163	163	163	154

iPhone 6	iPhone 6+	iPhone 6+ (a)	iPhone 6+ (b)	iPhone 6+ (c)
4.7"	5.5"	5.5"	5.5"	5.5"
750x1334	1080x1920 (downsampling)	1242x2208	1080x1920	1080x1920
~	1242x2208	~	~	~
375x667	414x736	414x736	360x640	540x960
@2x	@3x	@3x	@3x	@2x
326	401	461	401	401
163	154	154	134	200

iPhone 6 Plus

iPhone 6

iPhone 5 [5]



Retina HD 高清显示屏：

5.5 英寸 (对角线) LED 背光宽
Multi-Touch 显示屏，具有 IPS
技术

1920 x 1080 像素分辨率，
401 ppi

Retina HD 高清显示屏：

4.7 英寸 (对角线) LED 背光宽
Multi-Touch 显示屏，具有 IPS
技术

1334 x 750 像素分辨率，326 ppi

Retina 显示屏：

4 英寸 (对角线) LED 背光宽
Multi-Touch 显示屏，具有 IPS
技术

1136 x 640 像素分辨率，326 ppi

4.3 移动端适配方案

4.3.1 viewport 适配

1. 为什么做 viewport 适配
 - a) 手机厂商在生产手机的时候大部分手机默认页面宽度为 980px
 - b) 手机实际视口宽度都要小于 980px，如: iphone6 为 375px
 - c) 开发需求： 需要将 980 的页面完全显示在手机屏幕上且没有滚动条
2. 实现：

```
<meta name="viewport" content="width=device-width,initial-scale=1.0">
```

4.3.2 rem 适配

1. 为什么做 rem 适配
 - a) 机型太多，不同的机型屏幕大小不一样
 - b) 需求： 一套设计稿的内容在不同的机型上呈现的效果一致，根据屏幕大小不同的变化，页面中的内容也相应变化
2. 实现：

```
function remRefresh() {  
    let clientWidth = document.documentElement.clientWidth;  
    // 将屏幕等分 10 份  
    let rem = clientWidth / 10;  
    document.documentElement.style.fontSize = rem + 'px';  
    document.body.style.fontSize = '12px';  
}  
  
window.addEventListener('pageshow', () => {  
    remRefresh()  
})  
  
// 函数防抖  
let timeoutId;  
window.addEventListener('resize', () => {  
    timeoutId && clearTimeout(timeoutId);  
    timeoutId = setTimeout(() => {  
        remRefresh()  
    }, 300)  
})
```

3. 第三方库实现
[lib-flexible + px2rem-loader](#)

第 5 章 小程序特点

5.1 小程序特点概述

1. 没有 DOM
2. 组件化开发： 具备特定功能效果的代码集合
3. 体积小，单个压缩包体积不能大于 2M，否则无法上线
4. 小程序的四个重要的文件
 - a) *.js
 - b) *.wxml ----> view 结构 ----> html
 - c) *.wxss ----> view 样式 -----> css
 - d) *.json ----> view 数据 -----> json 文件
5. 小程序适配方案: rpx (responsive pixel 响应式像素单位)
 - a) 小程序适配单位: rpx
 - b) 规定任何屏幕下宽度为 750rpx
 - c) 小程序会根据屏幕的宽度不同自动计算 rpx 值的大小
 - d) Iphone6 下: $1\text{rpx} = 1 \text{ 物理像素} = 0.5\text{px}$

小程序编译后，rpx会做一次px换算。换算是以375个物理像素为基准，也就是在一个宽度为375物理像素的屏幕下， $1\text{rpx} = 1\text{px}$ 。

举个例子：iPhone6屏幕宽度为375px，共750个物理像素，那么 $1\text{rpx} = 375 / 750 \text{ px} = 0.5\text{px}$ 。

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

图2-11 常用机型rpx尺寸换算表

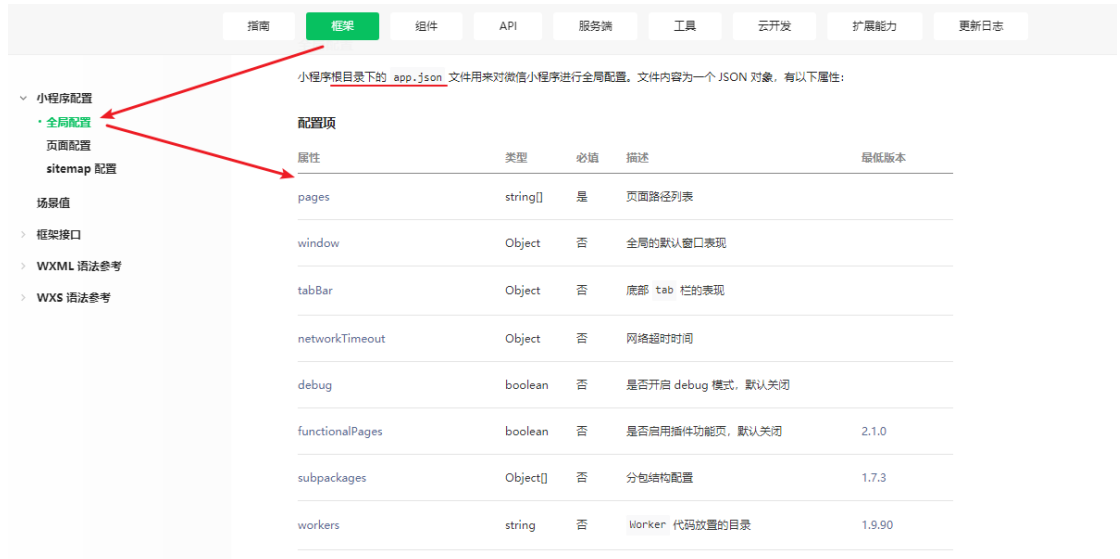
5.2 小程序配置

5.2.1 全局配置: app.json

1. 作用： 用于为整个应用进行选项设置
2. 链接：

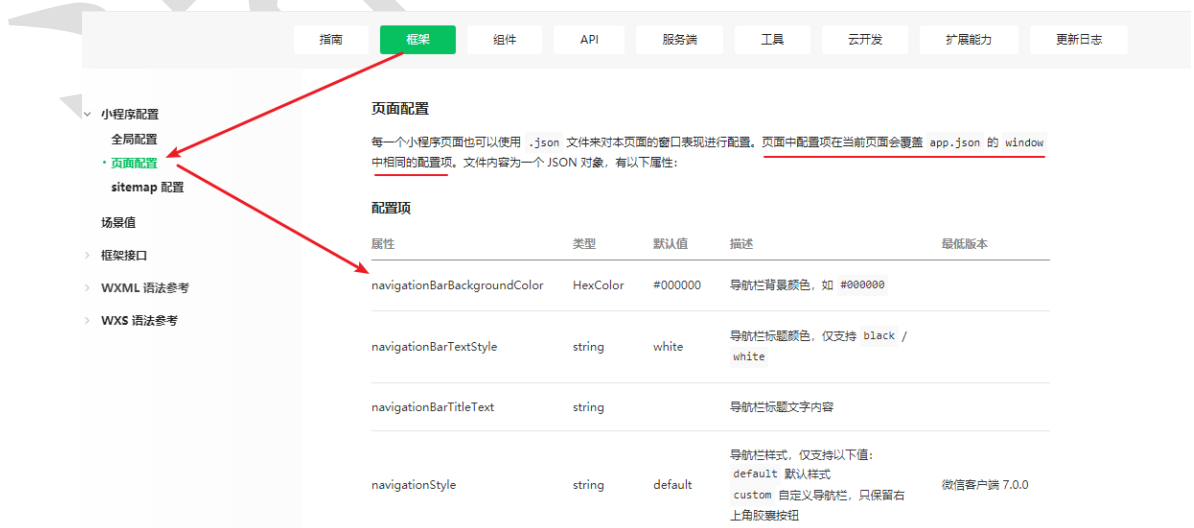
- a) <https://developers.weixin.qq.com/miniprogram/dev/reference/configuration/app.html>

3. 配图:



5.2.2 页面配置：页面名称.json

- 作用：用于为指定的页面进行配置
- 链接：
 - <https://developers.weixin.qq.com/miniprogram/dev/reference/configuration/app.html>
- 注意事项：页面配置的优先级高于全局配置
- 配图：



5.2.3 sitemap 配置：sitemap.json

1. 作用：用于被微信搜索爬取页面
2. 链接：
 - a) <https://developers.weixin.qq.com/miniprogram/dev/reference/configuration/app.html>
3. 配图：



5.3 小程序框架接口

5.3.1 App

1. 全局 app.js 中执行 App()
2. 生成当前应用的实例对象
3. getApp() 获取全局应用实例

5.3.2 Page

1. 页面.js 中执行 Page()
2. 生成当前页面的实例
3. 通过 getCurrentPages 获取页面实例

第 6 章 wxml 语法

6.1 数据绑定

6.1.1 初始化数据

1. 页面.js 的 data 选项中

```
Page({
  data: {
    message: 'Hello MINA!'
  }
})
```

6.1.2 使用数据

1. 模板结构中使用双大括号 {{message}}
2. 注意事项: 小程序中为单项数据流 model ---> view

```
<view> {{ message }} </view>
```

6.1.3 修改数据

1. this.setData({message: '修改之后的数据'}, callback)
2. 特点:
 - a) 同步修改: this.data 值被同步修改
 - b) 异步更新: 异步将 setData 函数用于将数据从逻辑层发送到视图层 (异步)

```
// index.js
Page({
  data: {
    text: 'init data',
    num: 0,
    array: [{text: 'init data'}],
    object: {
      text: 'init data'
    }
  },
  changeText: function() {
    // this.data.text = 'changed data' // 不要直接修改 this.data
    // 应该使用 setData
    this.setData({
      text: 'changed data'
    })
  },
})
```

6.2 事件绑定

6.2.1 事件分类

1) 冒泡事件

a) 定义：冒泡事件：当一个组件上的事件被触发后，该事件会向父节点传递。

b) 冒泡事件列表：

<https://mp.weixin.qq.com/debug/wxadoc/dev/framework/view/wxml/event.html>

2) 非冒泡事件

a) 定义：当一个组件上的事件被触发后，该事件不会向父节点传递。

b) 非冒泡事件：表单事件和自定义事件通常是非冒泡事件

<https://mp.weixin.qq.com/debug/wxadoc/dev/framework/view/wxml/event.html>

6.2.2 绑定事件

1. bind 绑定：事件绑定不会阻止冒泡事件向上冒泡

```
<view bindtap="handleTap" class='start_container'>
  <text class='start'>开启小程序之旅</text>
</view>
```

2. catch 绑定：事件绑定可以阻止冒泡事件向上冒泡

```
<view catchtap="handleTap" class='start_container'>
  <text class='start'>开启小程序之旅</text>
</view>
```

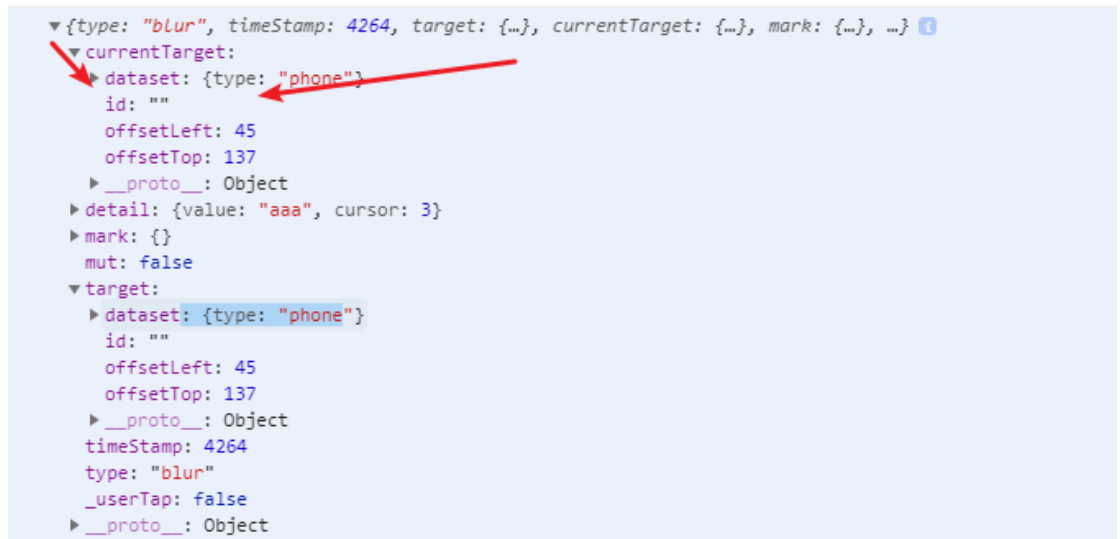
6.2.3 向事件对象传参

1. 语法：data-key=value

```
<text class="tit">手机号码</text>
<input bindblur="handleBlur" data-type="phone" type="text" placeholder="请输入手机号码"/>
</view>
<view class="input-item">
```

key value

2. 获取：event.target.dataset.key || event.currentTarget.dataset.key



3. Event.target 和 event.currentTarget 的区别
- Event.target 是触发事件的对象，但不一样是绑定事件的对象，如：事件委托，冒泡
 - currentTarget 触发时间的对象一定是绑定事件的对象，没有事件委托

6.3 列表渲染

6.3.1 语法说明

1. wx:for='{{arr}}'
2. wx:key='{{唯一值}}'

6.3.2 注意事项

1. 默认的个体: item
2. 默认的下标: index
3. 自定义个体变量名称: wx:for-item='myItem'
4. 自定义下标变量名称: wx:for-index='myIndex'

6.4 条件渲染

6.4.1 语法说明

1. wx:if='条件'
2. wx:elif='条件'

3. wx:else

wx:if

在框架中, 使用 `wx:if=""` 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` 来添加一个 else 块:

```
<view wx:if="{{length > 5}}"> 1 </view>
<view wx:elif="{{length > 2}}"> 2 </view>
<view wx:else> 3 </view>
```

6.4.2 wx:if VS hidden

1. `hidden` 用法: `<view hidden='{{true}}'></view>`
2. `wx:if` 等同于 `v-if`, 条件为 `false` 的时候不加载, 条件切换的时候决定元素销毁或者重新加载渲染
3. `hidden` 等同于 `v-show`, 始终加载元素, 条件切换的时候决定元素的显示和隐藏

6.5 模板使用

6.5.1 定义模板

定义模板

使用 `name` 属性, 作为模板的名字。然后在 `<template/>` 内定义代码片段, 如:

```
<!--
  index: int
  msg: string
  time: string
-->
<template name="msgItem">
  <view>
    <text> {{index}}: {{msg}} </text>
    <text> Time: {{time}} </text>
  </view>
</template>
```


6.5.2 引入模板

1. 引入模板结构: `<import src='模板结构相对路径' />`
2. 引入模板样式: `@Import '模板样式路径'`

6.5.3 使用模板

使用模板


使用 `is` 属性, 声明需要的使用的模板, 然后将模板所需要的 `data` 传入, 如:



```
<template is="msgItem" data="{{...item}}" />
```


6.5.4 向模板导入数据并使用数据

```
-->
<template name="msgItem">
  <view>
    <text> {{index}}: {{msg}} </text>
    <text> Time: {{time}} </text>
  </view>
</template>
```



使用模板

使用 `is` 属性, 声明需要的使用的模板, 然后将模板所需要的 `data` 传入, 如:



```
<template is="msgItem" data="{{...item}}" />
```

```
Page({
  data: {
    item: {
      index: 0,
      msg: 'this is a template',
      time: '2016-09-15'
    }
  }
})
```



6.6 生命周期

6.6.1 对应阶段说明

1. onLoad(Object query)

a) 页面加载时触发。一个页面只会调用一次，可以在 `onLoad` 的参数中获取打开当前页面路径中的参数。

b) **参数：**

名称	类型	说明
----	----	----

query	Object	打开当前页面路径中的参数
-------	--------	--------------

2. onShow()

a) 页面显示/切入前台时触发

b) 会执行多次

3. onReady()

a) 页面初次渲染完成时触发。一个页面只会调用一次，代表页面已经准备妥当，可以和视图层进行交互。

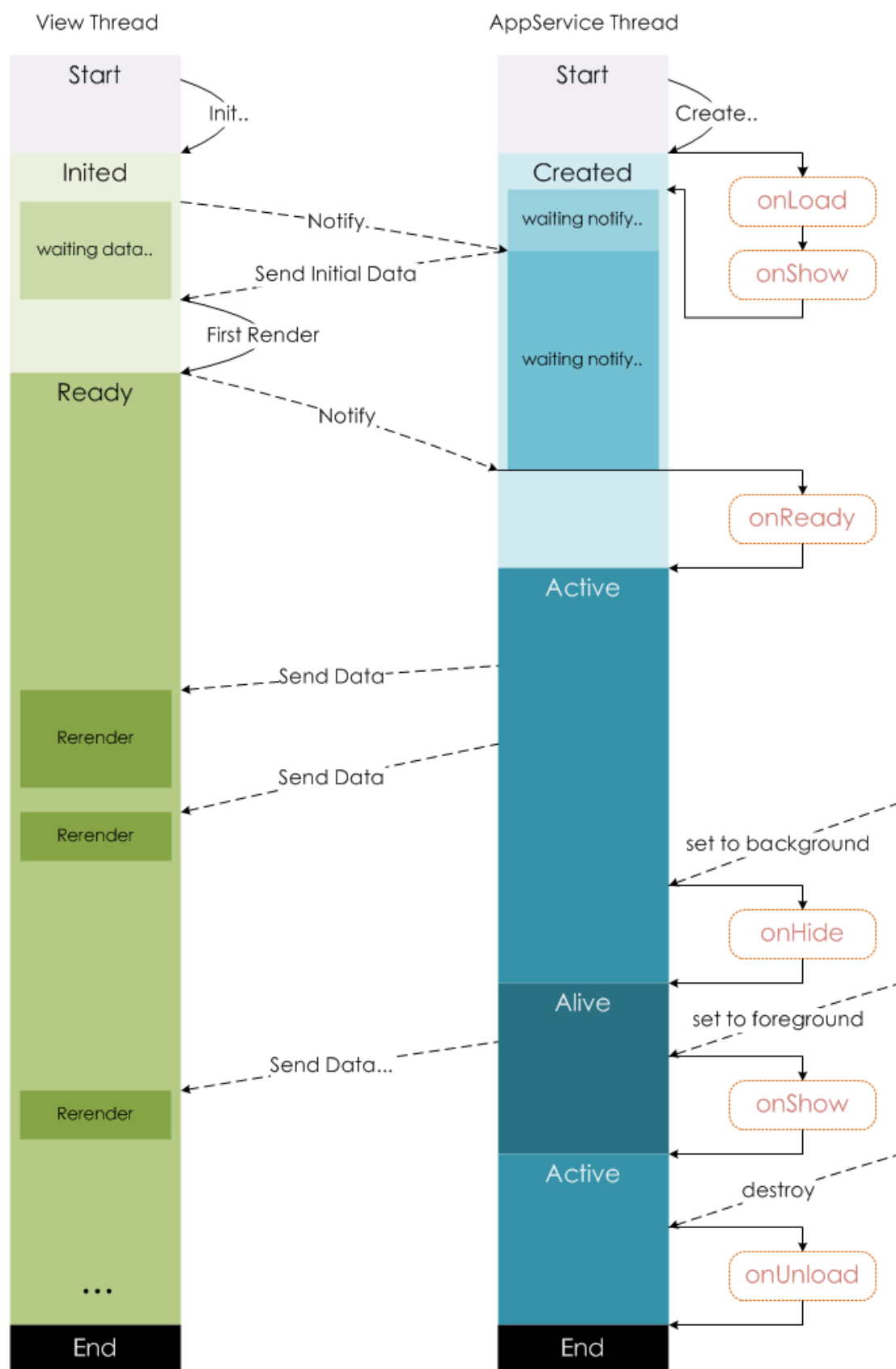
4. onHide()

a) 页面隐藏/切入后台时触发。如 `wx.navigateTo` 或底部 `tab` 切换到其他页面，小程序切入后台等。

5. onUnload()

a) 页面卸载时触发。如 `wx.redirectTo` 或 `wx.navigateBack` 到其他页面时。

6.6.2 官网图示说明



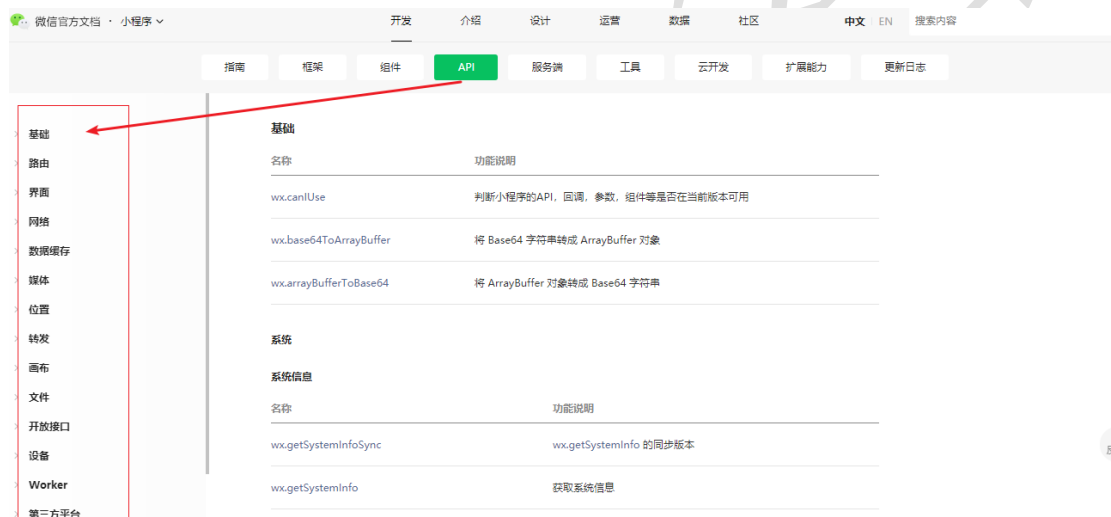
6.6.3 官网对应地址

<https://developers.weixin.qq.com/miniprogram/dev/framework/app-service/page-life-cycle.html>

第 7 章 小程序 API

7.1 API 使用说明

1. 小程序提供了很多实用的方法供开发者使用
2. 小程序全局对象是: `wx`
3. 所有的 API 都保存在 `wx` 对象中



7.2 常用 API

1. 界面交互
 - a) 显示消息提示框: `wx.showToast()`
 - b) 显示消息加载框: `wx.showLoading()`
 - c) 关闭消息提示框: `wx.hideToast()`
 - d) 关闭消息加载框: `wx.hideLoading()`
2. 路由跳转
 - a) `wx.navigateTo()`
 - b) `wx.redirectTo()`
 - c) `wx.switchTab()`

3. 网络请求
 - a) `wx.request()`
4. 本地存储
 - a) `wx.setStorage()`
 - b) `wx.setStorageSync()`
 - c) `wx.getStorage()`
 - d) `wx.getStorageSync()`
5. 媒体
 - a) `wx.getBackgroundAudioManager()`
 - b) `wx.playVoice()`

7.3 快速查找技巧

1. 小程序的初学者可能对于小程序的官网的众多内容一时毫无头绪，无从下手不知道从哪来找想要的内容
2. 当在小程序中想要实现某一种布局，查看：[组件](#)
3. 当在小程序中想要实现某一个功能，查看：[API](#)
4. 当在小程序中想要进行某一个配置或者某一种页面语法，查看：[框架 + 指南](#)
5. 查看小程序官网的时候要细心，最好是将要使用的 [API](#) 的相关内容看完整，因为 [API](#) 的配置及限制较多

第 8 章 小程序重点知识汇总

8.1 小程序本地存储

8.1.1 语法说明

1. 存入数据
 - a) `wx.setStorage()` 异步
 - b) `wx.setStorageSync()` 同步

```
wx.setStorage({
  key: "key",
  data: "value"
})
```

2. 读取数据
 - a) `wx.getStorage()` 异步

b) wx.getStorageSync() 同步

```
wx.getStorage({
  key: 'key',
  success (res) {
    console.log(res.data)
  }
})
```

3. 删除数据

a) wx.removeStorage() 异步

b) wx.removeStorageSync() 同步

```
wx.removeStorage({
  key: 'key',
  success (res) {
    console.log(res)
  }
})
```

4. 清空数据

a) wx.clearStorage() 异步

b) wx.clearStorageSync() 同步

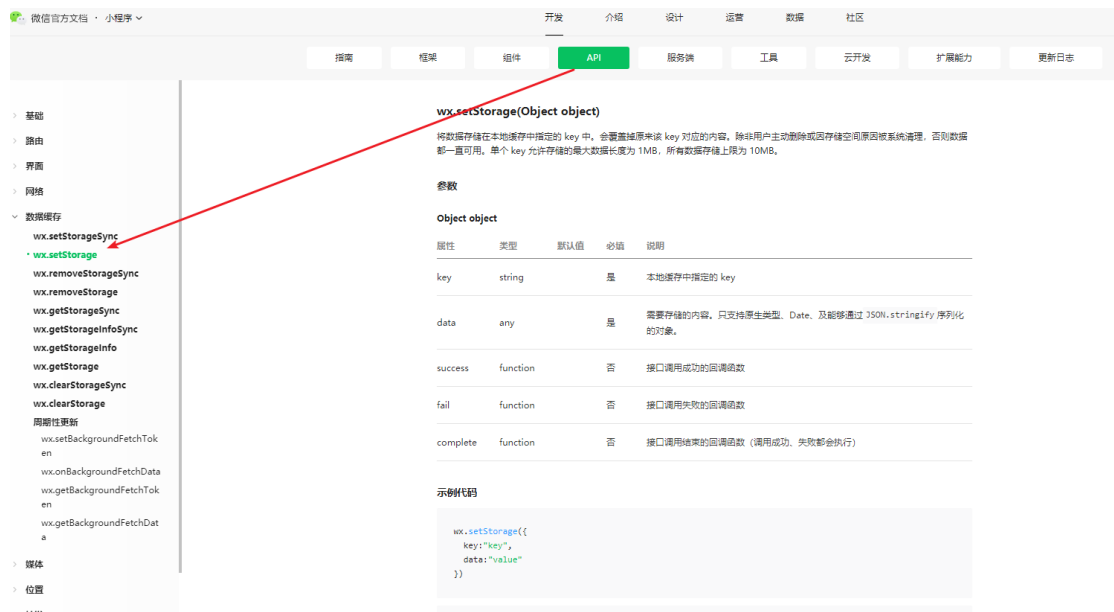
```
wx.clearStorage()
```

8.1.2 注意事项

1. 除非用户主动删除或因存储空间原因被系统清理，否则数据都一直可用
2. 单个 key 允许存储的最大数据长度为 1MB
3. 所有数据存储上限为 10MB

8.1.3 官网对应地址

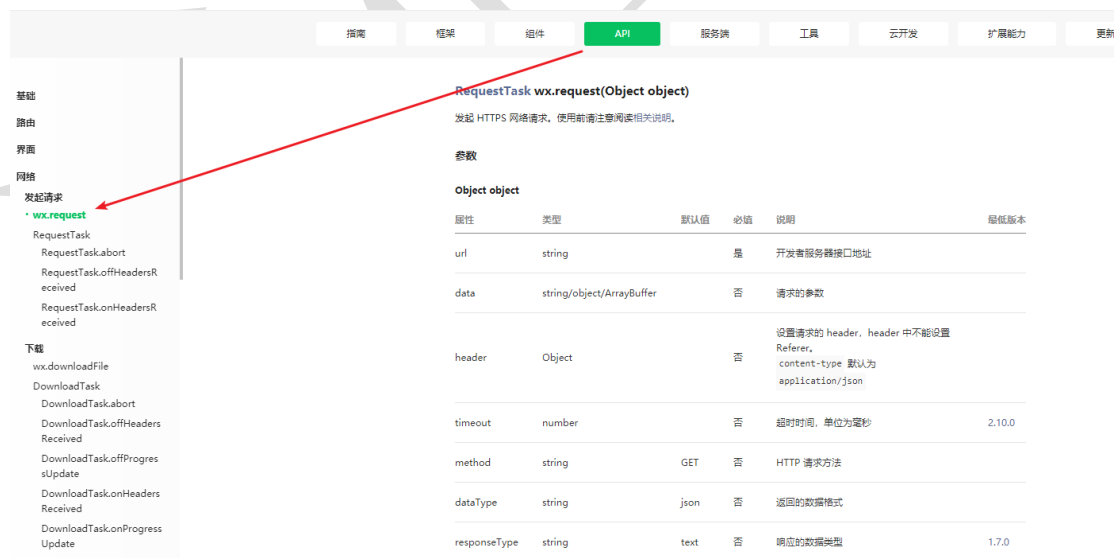
<https://developers.weixin.qq.com/miniprogram/dev/api/storage/wx.setStorage.html>



8.2 小程序前后端交互

8.2.1 语法说明

1. wx.request()



8.2.2 相关配置

1. 每个微信小程序需要事先设置通讯域名，小程序只可以跟指定的域名进行网络通

信

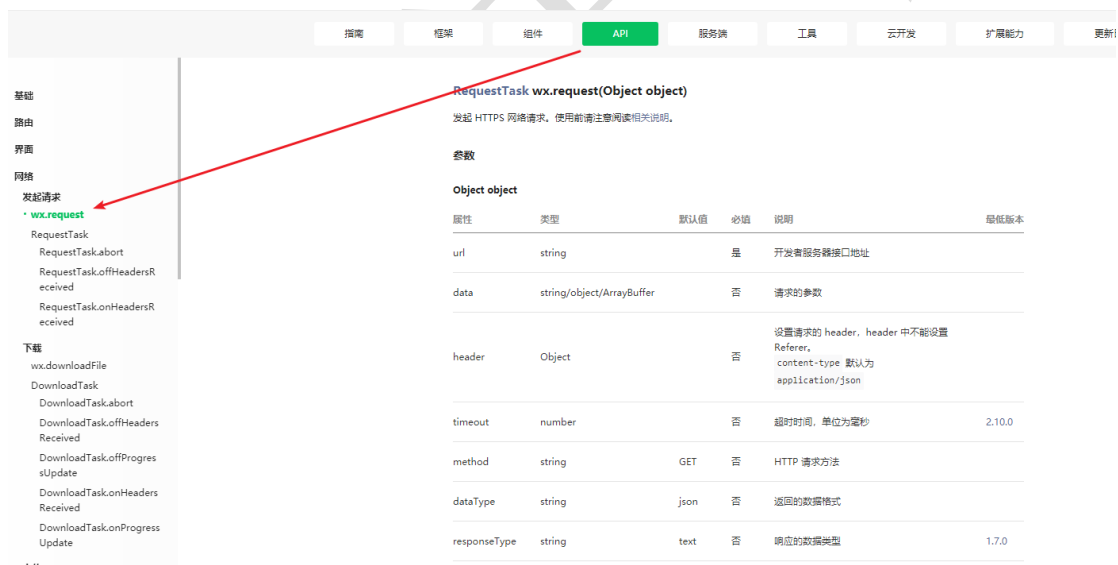
2. 服务器域名请在 「小程序后台-开发-开发设置-服务器域名」 中进行配置
3. 默认超时时间和最大超时时间都是 60s
4. 超时时间可以在 app.json 中通过 networktimeout 配置

8.2.3 注意事项

1. 小程序为了安全起见只支持 Https 请求
2. wx.request 最大并发限制 10 个

8.2.4 官网对应地址

<https://developers.weixin.qq.com/miniprogram/dev/api/storage/wx.setStorage.html>



The screenshot shows the WeChat Mini Program API documentation. The left sidebar has a menu with categories like '基础' (Basic), '路由' (Routing), '界面' (Interface), and '网络' (Network). Under '网络', there is a sub-menu '发起请求' (Initiate Request) which contains 'wx.request'. A red arrow points from this link to the main content area. The main content area shows the title 'RequestTask wx.request(Object object)' and a description '发起 HTTPS 网络请求。使用前请注意阅读相关说明。' (Initiate HTTPS network request. Please read the relevant documentation before use). Below this is a table of parameters for the 'Object object'.

属性	类型	默认值	必填	说明	最低版本
url	string		是	开发者服务器接口地址	
data	string/object/ArrayBuffer		否	请求的参数	
header	Object		否	设置请求的 header，header 中不能设置 Referer，content-type 默认为 application/json	
timeout	number		否	超时时间，单位为毫秒	2.10.0
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	1.7.0

8.3 小程序页面通信

8.3.1 路由传参

1. 传参方式
 - a) 路由地址中 + query 传参数
 - b) 示例： url?a=123

2. 获取参数

- a) 跳转目标页面的 onLoad 函数中的 options 实参中获取

```
/**
 * 生命周期函数--监听页面加载
 */
onLoad: async function (options) {
  // 获取路由传参数据: 音乐id
  let id = options.id*1; // 注意接收的
```

8.3.2 消息订阅发布

1. 使用第三方库: pubsub-js
2. 安装: npm install pubsub-js
3. 使用:
 - a) Import PubSub from 'pubsub-js'
 - b) 订阅消息: PubSub.subscribe('eventName', callback)
 - c) 发布消息: PubSub.publish('eventName', data)
 - d) 取消订阅: PubSub.unsubscribe('eventName')

8.3.3 eventChannel 事件通道

1. 订阅事件

- a) wx.navigateTo()跳转的时候在 events 选项中定义事件名及事件对应的回调

```
wx.navigateTo({
  url: 'test?id=1',
  events: {
    // 为指定事件添加一个监听器，获取被打开页面传送到当前页面的数据
    acceptDataFromOpenedPage: function(data) {
      console.log(data)
    },
    someEvent: function(data) {
      console.log(data)
    }
    ...
  },
  success: function(res) {
    // 通过eventChannel向被打开页面传送数据
    res.eventChannel.emit('acceptDataFromOpenerPage', { data: 'test' })
  }
})
```

在navigateTo的events选项中配置

自定义事件名

2. 获取事件总线对象

- a) 目标页面中通过: 实例.getOpenerEventChannel()

- b) 示例: `const eventChannel = this.getOpenerEventChannel()`
- 3. 触发事件
 - a) `eventChannel.emit('事件名', data)`

```
//test.js
Page({
  onLoad: function(option){
    console.log(option.query)
    const eventChannel = this.getOpenerEventChannel()
    eventChannel.emit('acceptDataFromOpenedPage', {data: 'test'});
    eventChannel.emit('someEvent', {data: 'test'});
    // 监听acceptDataFromOpenerPage事件，获取上一页面通过eventChannel传送到当前页面的数据
    eventChannel.on('acceptDataFromOpenerPage', function(data) {
      console.log(data)
    })
  }
})
```

跳转目标页面

获取事件总线对象

触发自定义事件

8.4 小程序自定义组件

8.4.1 创建组件

1. 开发工具中右键新建组件
2. 组件对应的 json 文件中设置: `component: true`



```
navHeader.json x
1 {
2   "component": true,
3   "usingComponents": {}
4 }
```

8.4.2 使用组件

1. 使用组件的页面的 json 文件中注册使用组件

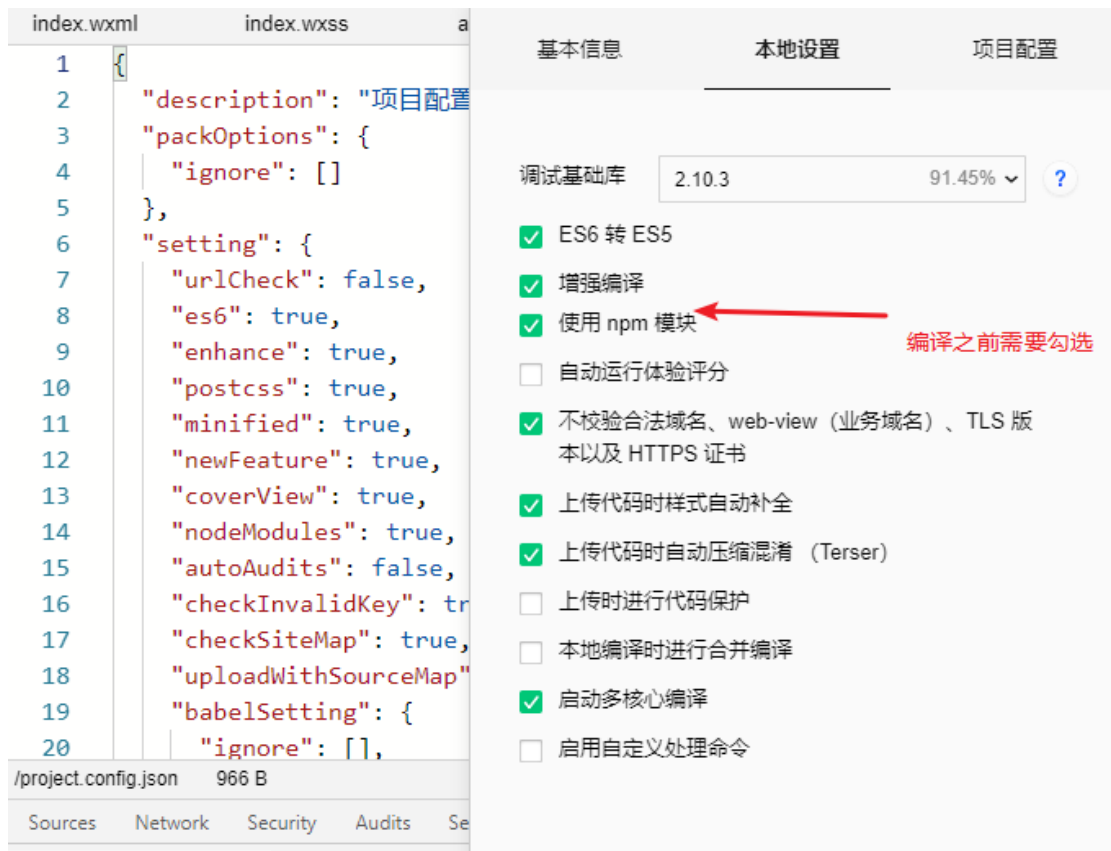


8.5 小程序使用 npm 包

8.5.1 初始化 package.json

npm init

8.5.2 勾选允许使用 npm



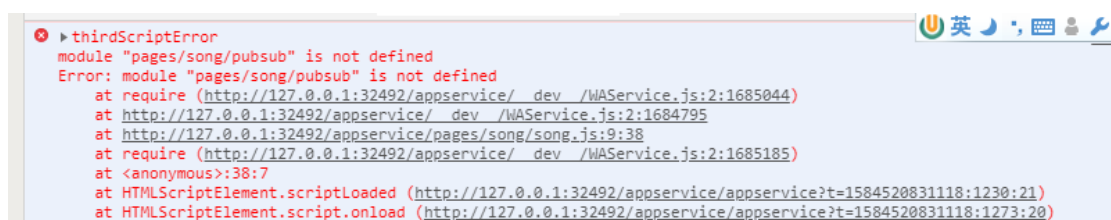
8.5.3 下载 npm 包

npm install packageName

8.5.4 构建 npm

1. 开发工具 ---> 工具 ---> 构建 npm
2. 会将 node_modules 中的包打包到 miniprogram_npm 中

8.5.5 流程执行不完整带来的错误

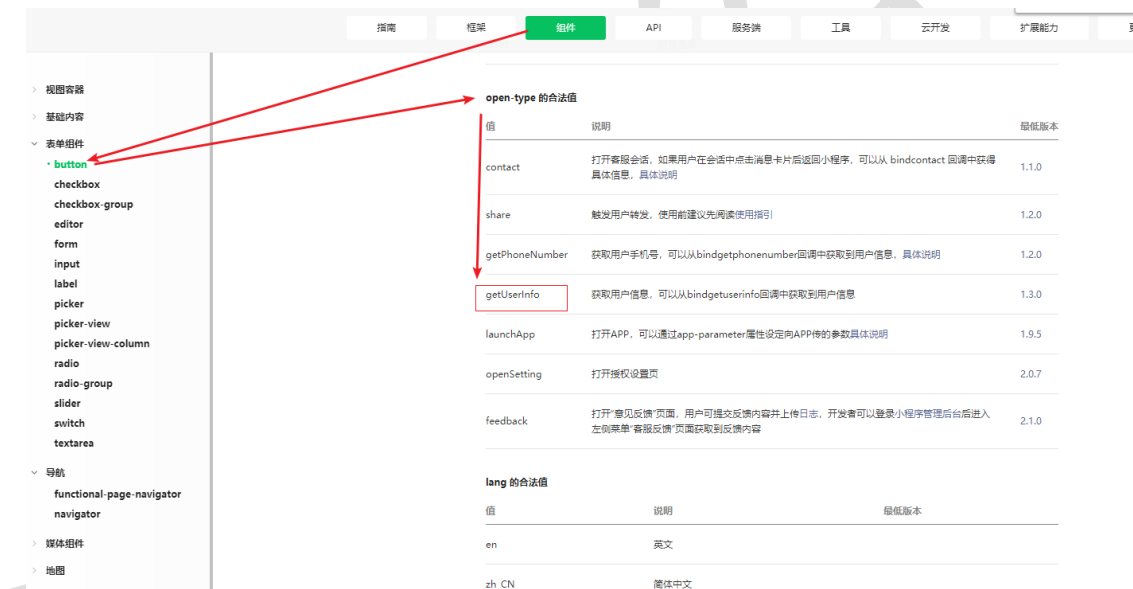


8.6 小程序获取用户基本信息

8.6.1 首次登陆获取

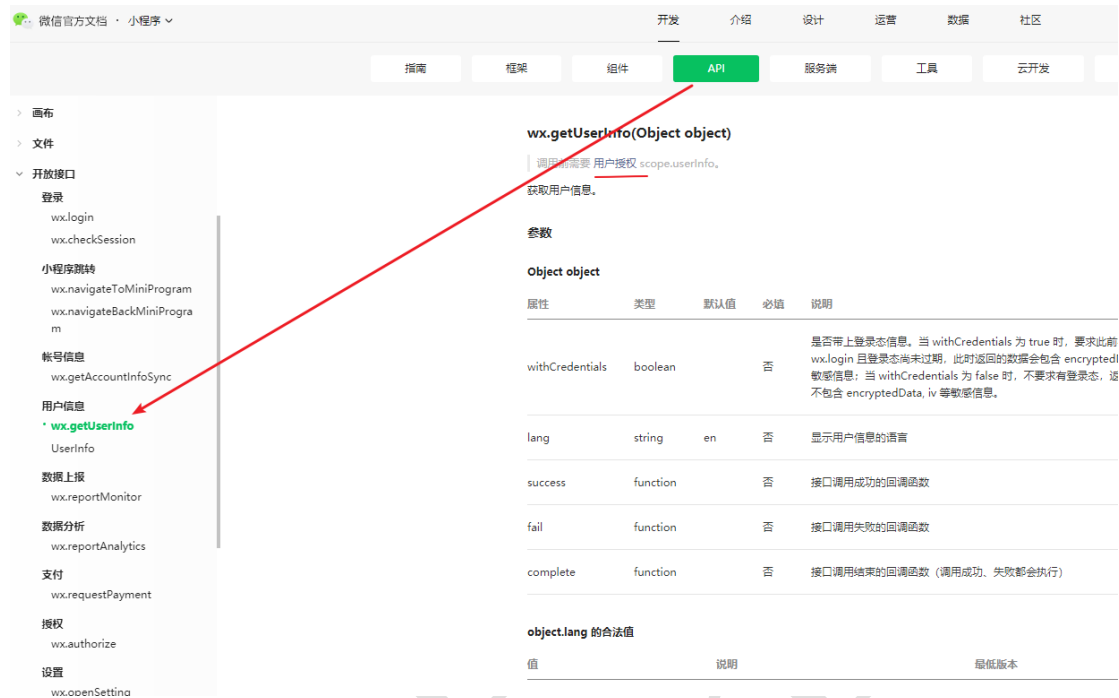
1. Button 组件设置 open-type 属性为 getUserInfo
2. `<button open-type='getUserInfo'></button>`
3. 设置后首次登陆点击 button 可以弹出授权窗口
4. 注意：授权的动作只发生一次，除非清除缓存，点击 button 授权一次之后再点击失效，不会弹出授权窗口
5. 官网对应地址

<https://developers.weixin.qq.com/miniprogram/dev/component/button.html>



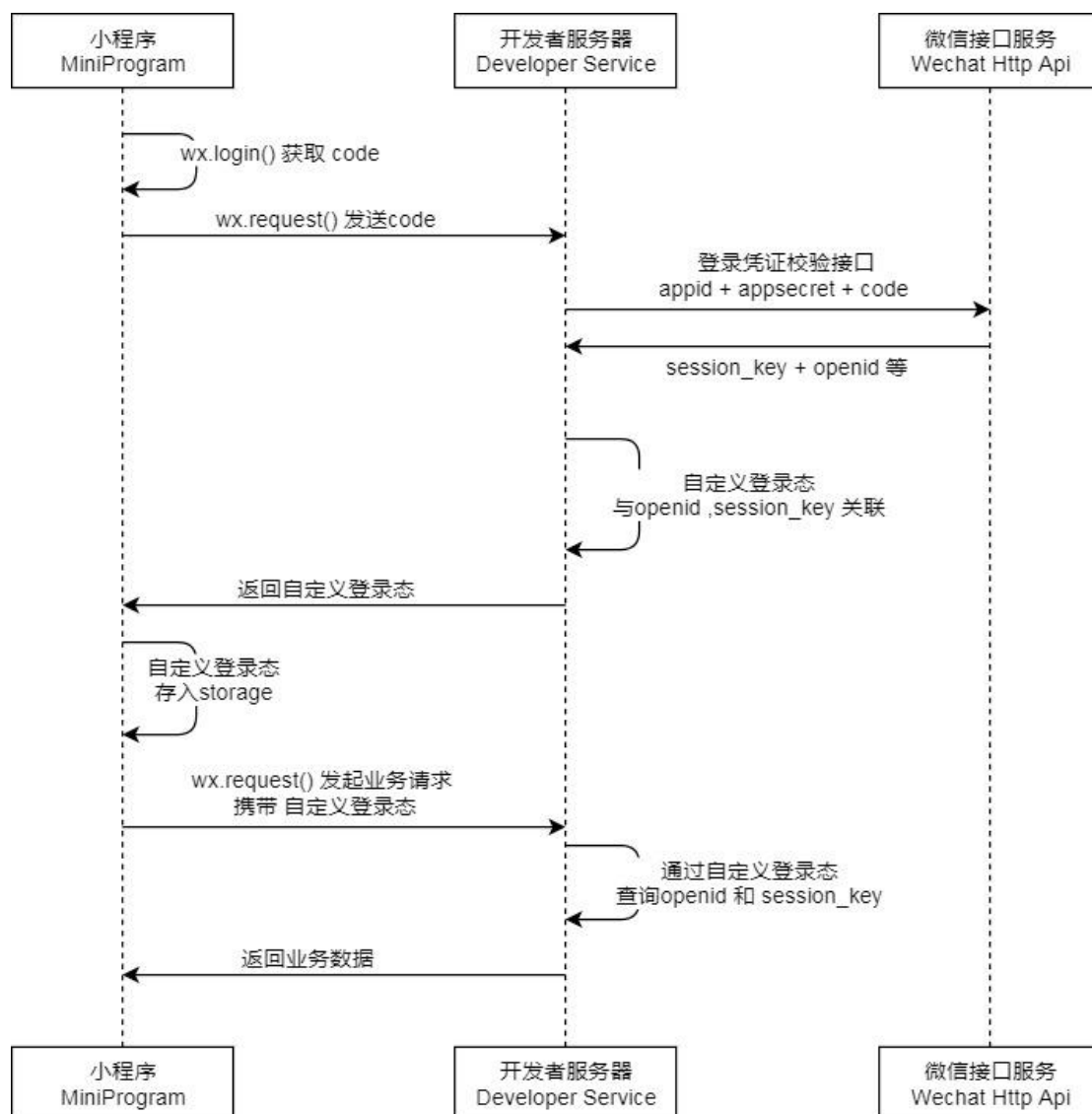
8.6.2 授权之后获取

1. wx.getUserInfo()
2. 官网对应地址:
<https://developers.weixin.qq.com/miniprogram/dev/api/open-api/user-info/wx.getUserInfo.html>



8.7 小程序获取用户唯一标识（openId）

8.7.1 官网图解




8.7.2 获取流程

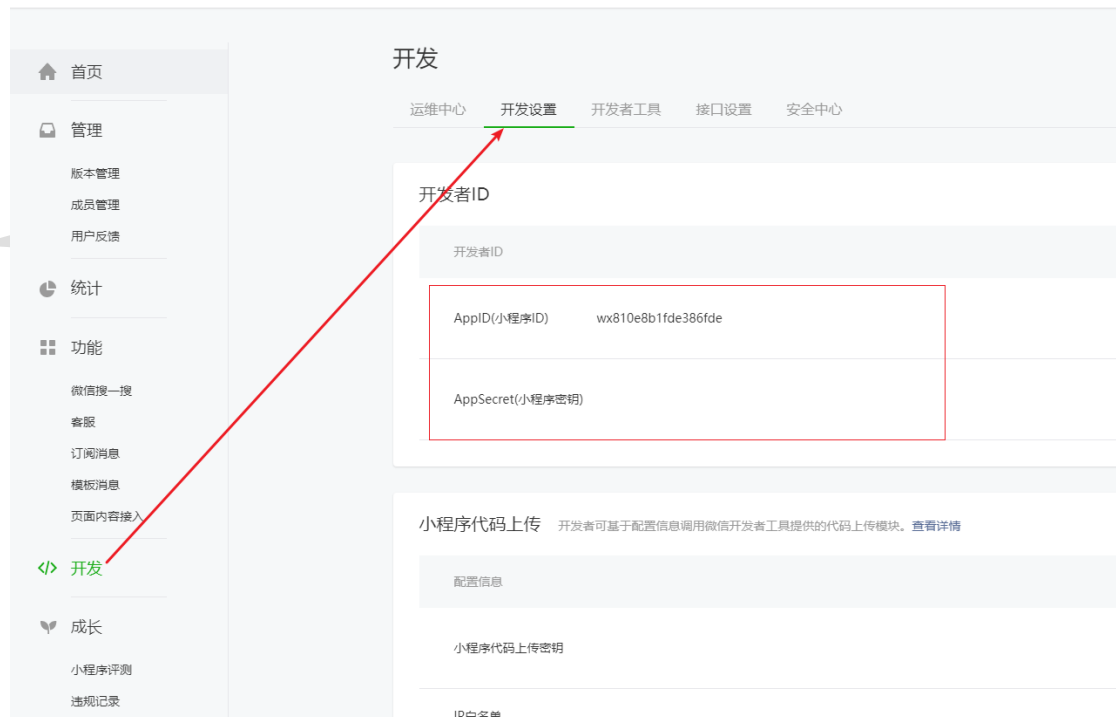
1. wx.login()

```
wx.login({
  success (res) {
    if (res.code) {
      //发起网络请求
      wx.request({
        url: 'https://test.com/onLogin',
        data: {
          code: res.code
        }
      })
    } else {
      console.log('登录失败!' + res.errMsg)
    }
  }
})
```

code为临时凭证

2. 发送 code 给服务器端
3. 服务器端发送请求携带参数(code, appSecret, appId)给微信服务器获取 openId
 - a) 接口地址:
4. appSecret, appId 在小程序首页获取

 小程序



5. 服务器获取 openId 后进行加密返回给前端

8.7.3 获取接口地址

请求方法： GET

```
https://api.weixin.qq.com/sns/jscode2session?appid=APPID&secret=SECR  
ET&js_code=JSCODE&grant_type=authorization_code
```

8.8 小程序分包流程

8.8.1 为什么要分包

1. 小程序要求压缩包体积不能大于 2M，否则无法发布
2. 实际开发中小程序体积如果大于 2M 就需要使用分包机制进行发布上传
3. 分包后可解决 2M 限制，并且能分包加载内容，提高性能
4. 分包后单个包的体积不能大于 2M
5. 分包后所有包的体积不能大于 16M

8.8.2 分包形式

1. 常规分包
2. 独立分包
3. 分包预下载

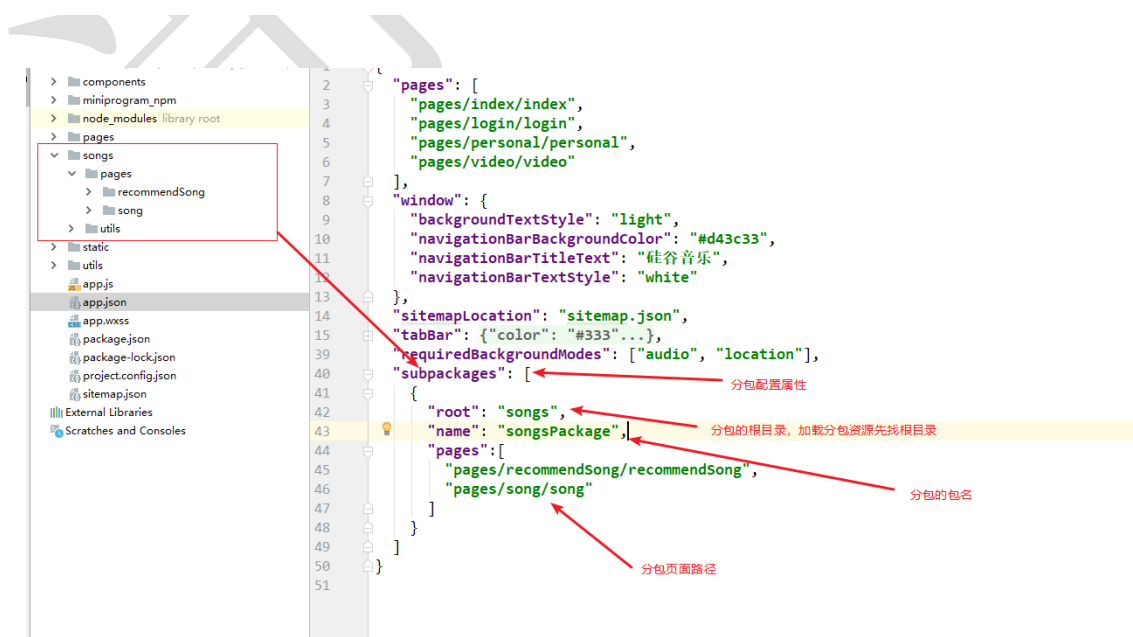
8.8.3 常规分包

1. 开发者通过在 `app.json subpackages` 字段声明项目分包结构
2. 特点：
 - a) 加载小程序的时候先加载主包，当需要访问分包的页面时候才加载分包内容
 - b) 分包的页面可以访问主包的文件，数据，图片等资源
 - c) 主包：
 - i. 主包来源：除了分包以外的内容都会被打包到主包中
 - ii. 通常放置启动页/tabBar 页面


```
{
  "pages": [
    "pages/index",
    "pages/logs"
  ],
  "subpackages": [
    {
      "root": "packageA",
      "pages": [
        "pages/cat",
        "pages/dog"
      ]
    }, {
      "root": "packageB",
      "name": "pack2",
      "pages": [
        "pages/apple",
        "pages/banana"
      ]
    }
  ]
}
```

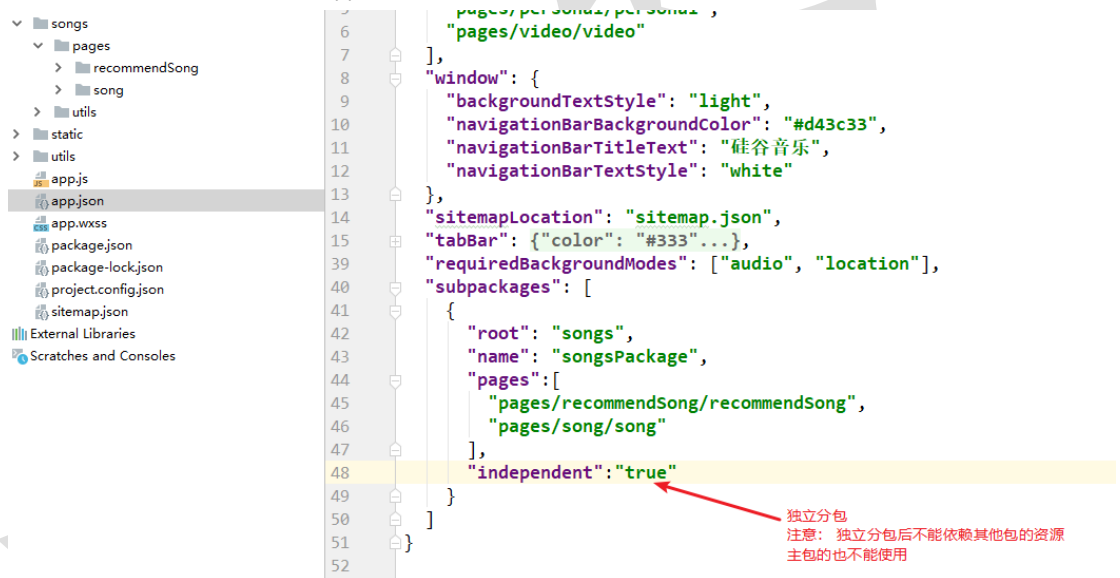
subpackages 中，每个分包的配置有以下几项：

字段	类型	说明
root	String	分包根目录
name	String	分包别名，分包预下载时可以使用
pages	StringArray	分包页面路径，相对与分包根目录
independent	Boolean	分包是否是独立分包



8.8.4 独立分包

1. 设置 independent 为 true
2. 特点:
 - a) 独立分包可单独访问分包的内容，不需要下载主包
 - b) 独立分包不能依赖主包或者其他包的内容
3. 使用场景
 - a) 通常某些页面和当前小程序的其他页面关联不大的时候可进行独立分包
 - b) 如：临时加的广告页 || 活动页



8.8.5 分包预下载

1. 配置
 - a) app.json 中设置 preloadRule 选项
 - b) key(页面路径): {packages: [预下载的包名 || 预下载的包的根路径]}

```

"preloadRule": {
  "pages/index": {
    "network": "all",
    "packages": ["important"]
  },
  "sub1/index": {
    "packages": ["hello", "sub3"]
  },
  "sub3/index": {
    "packages": ["path/to"]
  },
  "indep/index": {
    "packages": ["__APP__"]
  }
}

```



2. 特点:

- a) 在加载当前包的时候可以设置预下载其他的包
- b) 缩短用户等待时间，提高用户体验

8.8.6 分包效果演示



8.8.7 官网对应地址

<https://developers.weixin.qq.com/miniprogram/dev/framework/subpackages.html>

8.9 小程序转发分享

8.9.1 分享实现

1. Button 组件设置 open-type 为 share
2. `<button open-type='share' ></button>`

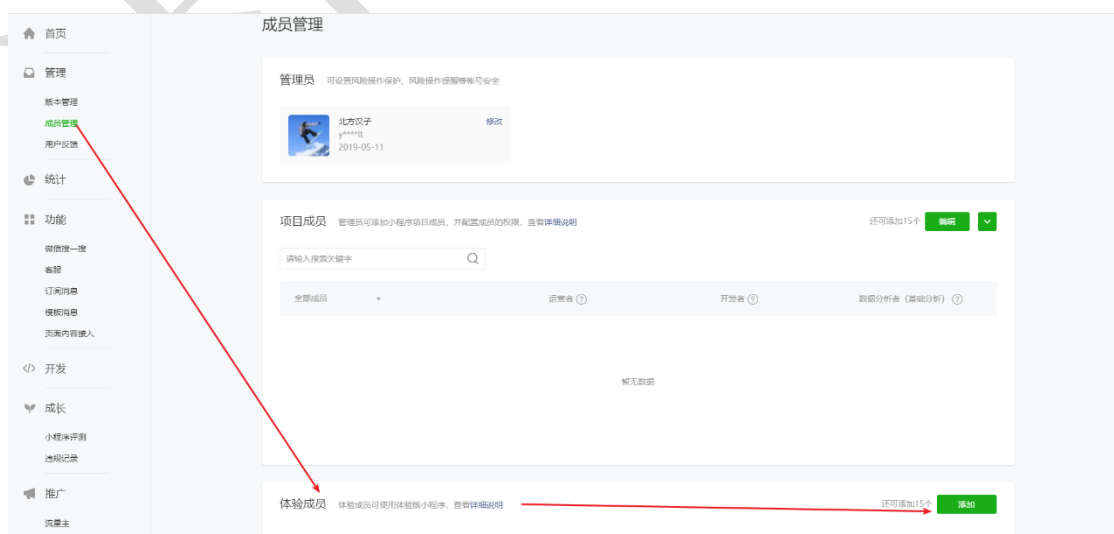
8.9.2 自定义分享内容

1. 生命周期回调中 onShareAppMessage 回调中 return 对象设置自定义内容

```
Page({
  onShareAppMessage: function (res) {
    if (res.from === 'button') {
      // 来自页面内转发按钮
      console.log(res.target)
    }
    return {
      title: '自定义转发标题',
      path: '/page/user?id=123'
    }
  }
})
```

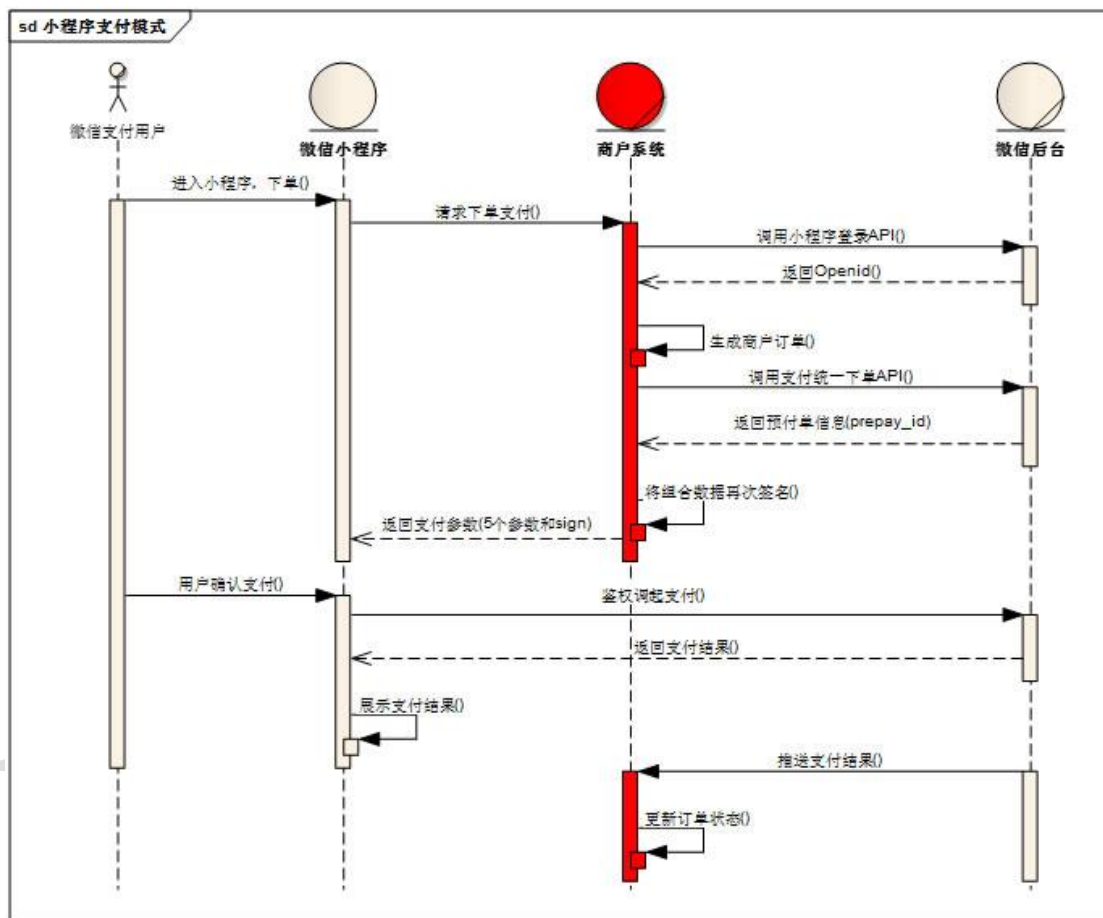
8.9.3 设置体验权限

1. 开发阶段分享给微信好友，默认没有体验权限，无法打开分享小程序，需要在开发页面设置
2. 最多添加 15 个微信好友



8.10 小程序支付流程

8.10.1 支付流程官网图解



8.10.2 支付流程详细说明

1. 用户在小程序客户端下单(包含用户及商品信息)
2. 小程序客户端发送下单支付请求给商家服务器
3. 商家服务器同微信服务器对接获取唯一标识 openid
4. 商家服务器根据 openid 生成商户订单(包含商户信息)
5. 商家服务器发送请求调用统一下单 API 获取预支付订单信息
 - a) 接口地址: <https://api.mch.weixin.qq.com/pay/unifiedorder>
6. 商家对预支付信息签名加密后返回给小程序客户端

- a) 签名方式: MD5
- b) 签名字段: 小程序 ID, 时间戳, 随机串, 数据包, 签名方式
- c) 参 考 地 址 :
https://pay.weixin.qq.com/wiki/doc/api/wxa/wxa_api.php?chapter=7_7&index=3
- 7. 用户确认支付 (鉴权调起支付)
 - a) API: wx.requestPayment()
- 8. 微信服务器返回支付结果给小程序客户端
- 9. 微信服务器推送支付结果给商家服务器端

8.10.3 官网对应地址

https://pay.weixin.qq.com/wiki/doc/api/wxa/wxa_api.php?chapter=7_3&index=1