微信小程序

目录

[微信小程序 1](#_Toc18788)

[一、 基础小程序组件 1](#_Toc27396)

[二、 小程序常用的部分组件 3](#_Toc11070)

[三、 小程序中的语法 5](#_Toc6702)

[四、 小程序自定义组件 9](#_Toc8625)

# 基础小程序组件

* 1. 小程序结构目录

|  |  |  |
| --- | --- | --- |
| 结构 | 传统web | 微信小程序 |
| 结构 | HTML | WXML |
| 样式 | CSS | WXSS |
| 逻辑 | Javascript | Javascript |
| 配置 | 无 | JSON |

* + - 1. 传统的web 其实和小程序的区别并不大
      2. 通过对比,可得出传统的web 是三层结构,而微信小程序是四层结构,多了一层配置.Json
  1. 基本的项目结构
     1. 
     2. 注: 页面文件可以采用分包机制进行项目分包
     3. <https://developers.weixin.qq.com/miniprogram/dev/framework/subpackages/basic.html>
  2. 小程序的配置文件
     1. 小程序分为两种配置文件, 一种是全局配置文件 app.json 和 页面的 index.json文件
        1. 注意: 配置文件中不能出现注释
           1. 全局配置 JSON 文件
        2. App.json 是当前小程序的全局配置, 包括了小程序的所有页面路径, 界面表现 , 网络超时时间,底部tab等
           1. 全局配置 JSON 文件 ( 右图==> )
        3. 字段的含义
           1. Pages 字段--- 用于描述当前小程序所有页面路径, 这是为了让微信客户端知道当前你的小程序定义在哪一个页面
           2. Window 字段 --- 定义小程序所有页面的顶部背景颜色,文字颜色定义等
           3. tabBar 字段 --- 底部tab 栏
  3. 小程序的页面配置文件
     1. Pages.json
     2. 这里的app.json是用来表示该小程序页面相关的配置
     3. 页面的配置只能设置 app.json 中部分window配置项的内容, 页面配置会覆盖app.json 的window 中相同的配置项

|  |  |  |  |
| --- | --- | --- | --- |
| **属性** | **类型** | **默认值** | **描述** |
| 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 |
| navigationBarTitleText | String |  | 导航栏标题文字内容 |
| backgroundColor | HexColor | #ffffff | 窗口的背景色 |
| backgroundTextStyle | String | dark | 下拉loading的样式, 仅支持dark / light |
| enablePullDownRefresh | Boolean | False | 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距离,单位px |

* 1. 小程序中最常使用的接口
     1. setData
        1. setData 函数用于将数据从逻辑层发送到视图层（异步），同时改变对应的 this.data 的值（同步）。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 必填 | 描述 | 最低版本 |
| data | Object | 是 | 这次要改变的值 |  |
| callback | Function | 否 | setData引起的界面更新渲染完毕后的回调函数 | 1.5.0 |

* + - 1. Object 以 key: value 的形式表示，将 this.data 中的 key 对应的值改变成 value;
         1. 其中 key 可以以数据路径的形式给出，支持改变数组中的某一项或对象的某个属性，如 array[2].message，a.b.c.d，并且不需要在 this.data 中预先定义。
      2. 注意点
         1. 直接修改this.data，而不调用this.setData()，是无法改变当前页面的状态的，会导致数据不一致
         2. 仅支持可以JSON化的数据
         3. 单次设置的数据不能超过1024KB，尽量避免一次设置过多的数据
         4. 如果在 onReady() 函数中 调用setData() 方法, 那么通过这个方法设置的值只能刷新一次, 若还需要再次刷新, 得 请清除缓存;
         5. onReady() 是 页面解析完成之后执行  
            官网地址:

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

# 小程序常用的部分组件

1. 常用的布局组件

View , text , rich-text, button ,image , icon , swiper , ridio , checkbox 等

* 1. View
     1. 视图容器 ---- 相当于之前的 div
     2. 属性

<view hover-class="h-class">

点击我试试

</view>

* 1. Text
     1. 文本标签 --- 相当于之前的span
  2. Icon
     1. 图标组件

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性 | 类型 | 默认值 | 必填 | 说明 |
| Type | String |  | 是 | Icon的类型,有效值: success, success\_no\_circle,info,warn,waiting,cancel,download,search, clear |
| Size | Number/string | 23 | 否 | Icon的大小 |
| Color | String |  | 否 | Icon的颜色, 同css的color |

* 1. Image
     1. 图片标签
        1. Image 组件默认宽高320\*240

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 类型 | 默认值 | 说明 |
| src | String |  | 图片资源地址 |
| mode | String | ‘scaleToFill’ | 图片剪切,缩放的模式 |

* + 1. Mode 有效值

|  |  |
| --- | --- |
| 值 | 说明 |
| scaleToFill | 缩放模式,不保持纵横缩放图片,使图片的宽高完全拉伸至填满image元素 |
| aspectFit | 缩放模式,保持纵横缩放图片,使图片的长边完能完全显示出来,也就是说,可以完整的将图片显示出来 |
| aspectFill | 缩放模式,保持纵横缩放图片,只保证图片的短边能完全显示出来,也即是, 图片通常只在水平或垂直方向是完整的,另一个将会发生截取 |

* 1. Swiper 轮播图组件

默认宽度100% .,高度150px

|  |  |  |  |
| --- | --- | --- | --- |
| 属性名 | 类型 | 默认值 | 说明 |
| Inducator-dots | Boolean | False | 是否显示面板指示点 |
| indicator-color | Color | Rgba(0,0,0,\,0.5) | 指示点颜色 |
| indicator-active-color | Color | #000000 | 当前选中的指示点颜色 |
| autoplay | Boolean | Fasle | 是否自动播放 |
| Interval | Number | 5000 | 自动切换时间间隔 |
| clrcular | Boolean | False | 是否循环轮播 |

* 1. button

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性 | 类型 | 默认值 | 必填 | 说明 |
| Size | String | Default | 否 | 按钮大小 |
| Type | String | Defalut | 否 | 按钮的样式 |
| Plain | Boolean | False | 否 | 按钮是否镂空.背景透明 |
| Disabled | Boolean | False | 否 | 是否禁用 |
| Loading | Boolean | False | 否 | 名称前是否带有loading图标 |
| Form-type | String | False | 否 | 用于from组件的submit(提交哦),reset(重置) |

* + 1. Size 的合法值

|  |  |
| --- | --- |
| 值 | 说明 |
| Default | 默认大小 |
| Mini | 小尺寸 |

* + 1. Type的合法值

|  |  |
| --- | --- |
| 值 | 说明 |
| Primary | 绿色 |
| Default | 白色 |
| Warn | 红色 |

# 小程序中的语法

<https://developers.weixin.qq.com/miniprogram/dev/reference/wxml/data.html>

(一): 数据绑定

1. WXML 中的动态数据均来自对应 Page 的 data;
2. 数据绑定使用 Mustache 语法（双大括号）将变量包起来
3. 组件属性 , 控制属性, 关键字, 三运运算 , 算术运算需要在双引号之间
4. : 列表渲染
   1. wx:for
      1. 遍历简单类型数组 / 对象数组
         1. 在组件上使用 wx:or 控制属性绑定一个数组, 即可使用数组中各项的数据重复渲染该组件. 默认数组的当前项的下标变量名默认为 index , 数组当前项的变量名默认为item
         2. 起别名 wx:for-item =’it ’ wx:for-index = ‘i’
         3. wx:key ( 建议每一个for 循环, 都写上 wx:key , 减少渲染次数,提升效率 )
         4. <https://developers.weixin.qq.com/miniprogram/dev/reference/wxml/list.html>
   2. 条件渲染
      1. Wx:if
         1. 在框架中使用 wx-if,来判断是否需要渲染该块代码
            1. 也可以使用wx:elif 和 wx:else 来添加一个else 块

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

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

1. wx:hidden
   1. 使用hidden , 该组件始终会被渲染,只是简单的控制显示和隐藏
   2. 注意:
      1. v-if 和 v-show 有本质上的区别
      2. 因为v-if 之中的模板也可能包含数据绑定, 所以当v-if 的条件值切换时,框架有一个局部渲染的过程,,因为会确保条件块在切换时销毁或重新渲染
      3. 同时 v-if 也是有惰性的 , 当条件为false时 ,它会什么也不做,只有当为true 时才会进行局部渲染
   3. 建议:
      1. 在一般情况下 wx-if 消耗的性能比hidden 更大, 因此在频繁切换的时候用hidden 更好, 反之, 则用if 更好;
2. : 引用
   1. WXML提供两种文件引入方式 import 和include
      1. Import
         1. Import可以在该文件中使用目标文件定义的 template , 如
            1. 在item.wxml zh9ong定义一个叫item 的template

<template name="item">

<text>{{text}}</text>

</template>

* + - * 1. <https://developers.weixin.qq.com/miniprogram/dev/reference/wxml/import.html>
      1. I
      2. include
         1. include可以将目标文件除了 <template /> < wxs/>外的整个代码引入,相当于是拷贝到 include 位置;

1. : 路由跳转
   1. wx:switchTab ( object, object )
      * + 1. 跳转到 tabBar 页面，并关闭其他所有非 tabBar 页面

wx.switchTab({

url: '/index'})

* 1. wx:reLaunch ( object, object )
     1. 关闭所有页面,打开到引用的某个页面
  2. wx.redirectTo( object, object )
     1. 关闭当前页面 跳转到应用内的某个页面。但是不允许跳转到 tabbar 页面
  3. wx.navigateBack(object, object)
     1. 保留当前页面，跳转到应用内的某个页面。但是不能跳到 tabbar 页面。使用 [wx.navigateBack](https://developers.weixin.qq.com/miniprogram/dev/api/route/wx.navigateBack.html) 可以返回到原页面。小程序中页面栈最多十层。
  4. 参数说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性 | 类型 | 默认值 | 必填 | 说明 |
| Url | String |  | 是 | 需要跳转的tabBar页面路径(不能携带参数) |
| Success | Function |  | 否 | 接口调用成功后的回调函数 |
| Fail | Function |  | 否 | 接口调用失败的回调函数 |
| Complete | Function |  | 否 | 接口调用结束后的回调函数( 不论失败成功都会执行) |

1. : 数据缓存
   1. wx.getStorageSync(string key)
      1. 从本地缓存中异步获取指定 key 的内容
   2. wx.setStorageSync(string key)
      1. 将数据存储在本地缓存中指定的 key 中。会覆盖掉原来该 key 对应的内容。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。单个 key 允许存储的最大数据长度为 1MB，所有数据存储上限为 10MB。===> 同步方法
   3. 参数 ( 注: wx.getStorageSync中没有Date )

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性 | 类型 | 默认值 | 必填 | 说明 |
| Key | String |  | 否 | 本地缓存中指定的key |
| Data | Any |  | 是 | 需要储存的内容,(只支持原生类型,Date,能够通过JSON,stringify序列化的对象) |
| 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 | 接口调用结束后的回调函数( 不论失败成功都会执行) |

实例

wx.getStorageSync({

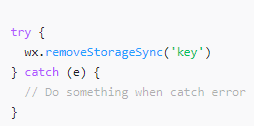
key: 'key',

success (res) {

console.log(res.data)

}})

* 1. wx.removeStorageSync(string key)
     1. 从本地缓存中移除指定 key



官方文档地址 <https://developers.weixin.qq.com/miniprogram/dev/api/storage/wx.setStorageSync.html>

1. : 数据缓存
   1. 在小程序发送请求
      1. wx.request(Object object)
         1. 常用的几个属性

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 | 请求的参数 |
| Header | Object |  | 否 | 设置的请求头 (可以不写) |
| Methid | String | GET | 否 | HTTP请求方式 |
| dataType | String | JSON | 否 | 返回的数据格式 |
| 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 |

Page({

// 绑定的事件

handleInput: function(e) {

e.detail.value

}

})

1. : 事件绑定
   1. bIndinput 与 bindtap
      1. Bindinput事件
         1. <input bindinput="handleInput" />
      2. bindTap 事件
         1. 绑定函数
            1. <button bindtap="click1">按钮1</button>
         2. 事件处理函数
            1. Page({
            2. click1() {
            3. console.log(111)
            4. }
            5. })
         3. 传参
            1. // 传递
            2. <button bindtap="click2" data-msg="abc" data-index="123">按钮2</button>
            3. // 接收
            4. click2(e){
            5. e.currentTarget.dataset
            6. },
2. 注
   1. 其实在小程序中的事件绑定就是在其前加一个bind 跟 vue 在前面加一个@ 相似
   2. 官方文档
      1. <https://developers.weixin.qq.com/miniprogram/dev/api/route/wx.switchTab.html>

# 小程序自定义组件

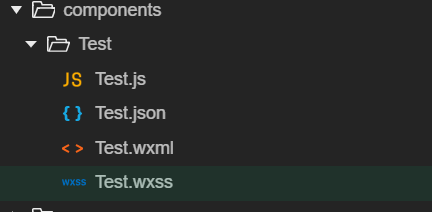
1. : 创建自定义组件
   1. 类似于页面，一个自定义组件由 json wxml wxss js 4个文件组成。要编写一个自定义组件，首先需要在 json 文件中进行自定义组件声明（将 component 字段设为 true 可将这一组文件设为自定义组件）
   2. 在组件的 json文件中进行自定义组件声明

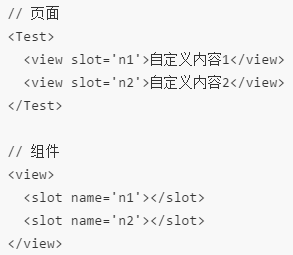
// Test.json

{

"component": true // 声明组件

}

* 1. <https://developers.weixin.qq.com/miniprogram/dev/framework/custom-component/>
  2. 然后创建文件夹 components/
  3. 创建组件 文件夹 Test/
  4. 新建 component

1. 引入组件
   1. 首先要在页面的json 文件中进行引用声明, 还需提供对应的组件名和组件路径
      1. // index.wxml 想使用组件的话, 就要在 index.json 里 先引入
      2. {
      3. "usingComponents": {
      4. // 要使用的组件的名称 // 组件的路径
      5. "Test":"/components/Test/Test"
      6. }
      7. }
   2. 使用组件
      1. <Test></Test>
   3. 使用solt
      1. 单个 直接在组件中 写上 <solt><solt/>
         * 1. // 页面 ====> 右图
           2. <Test>
           3. <view>自定义内容</view>
           4. </Test>
           5. // 组件
           6. <view>
           7. <view>前面的内容</view>
           8. <slot></slot>
           9. <view>后面的内容</view>
           10. </view>
      2.  多个solt -具名
         * 1. // 页面 ===> 右图
           2. <Test>
           3. <view slot='n1'>自定义内容1</view>
           4. <view slot='n2'>自定义内容2</view>
           5. </Test>
           6. // 组件
           7. <view>
           8. <slot name='n1'></slot>
           9. <slot name='n2'></slot>
           10. </view>
2. 组件通讯
   1. 父传子 : 父组件通过属性传递给子组件数据
      1. 代码如下
         * 1. // 父
           2. // 第一步 :父传子
           3. <Test msg='父的数据'></Test>
           4. // 子
           5. // 第二步 : 声明
           6. properties: {
           7. msg: {
           8. type: String,
           9. value: ''
           10. }
           11. },
           12. // 第三步 : 使用
           13. // 使用1 : 读取显示
           14. <view> 父传子 : {{ msg }} </view>
           15. // 使用2 : 打印
           16. attached() {
           17. console.log(123, this.properties.msg)
           18. }
   2. 子传父 : 子组件通过事件传递给父组件数据
      1. 代码如下
         1. // 父
         2. // 第一步 :父传子
         3. <Test msg='父的数据'></Test>
         4. // 子
         5. // 第二步 : 声明
         6. properties: {
         7. msg: {
         8. type: String,
         9. value: ''
         10. }
         11. },
         12. // 第三步 : 使用
         13. // 使用1 : 读取显示
         14. <view> 父传子 : {{ msg }} </view>
         15. // 使用2 : 打印
         16. attached() {
         17. console.log(123, this.properties.msg)
         18. }
3. 数据监听
   1. 数据监听器可以用于监听和响应任何属性和数据字段的变化。从小程序基础库版本从 2.6.1 开始支持。
      1. 使用监听数据
         1. 在一些情况下, 我们需要在一段数据字段被setData 设置时,需要执行一些操作
         2. 例如， this.data.sum 永远是 this.data.number  与 this.data.sumberB 的和。此时，可以使用数据监听器进行如下实现

Component({

attached: function() {

this.setData({

numberA: 1,

numberB: 2,

})

},

observers: {

'numberA, numberB': function(numberA, numberB) {

// 在 numberA 或者 numberB 被设置时，执行这个函数

this.setData({

sum: numberA + numberB

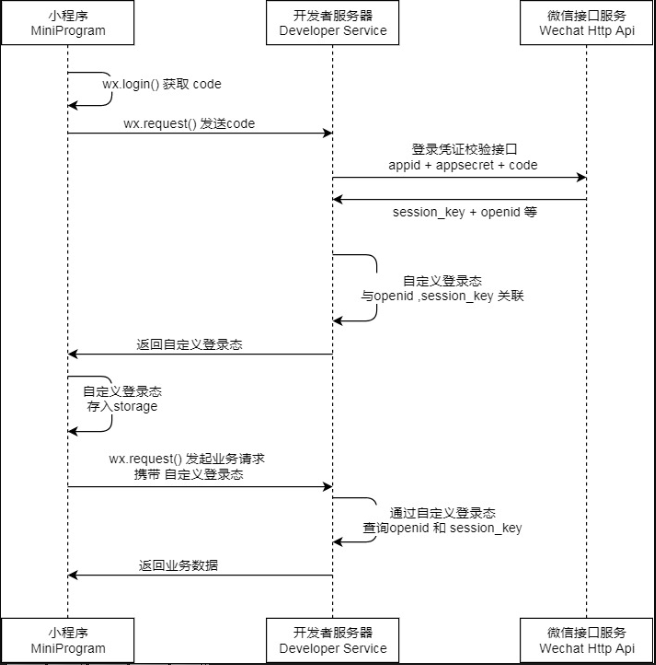
})

}

}})

* + 1. 文档地址:
    2. <https://developers.weixin.qq.com/miniprogram/dev/framework/custom-component/observer.html>

1. 小程序登录
   1. 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。



1. 调用 wx.login()获取 **临时登录凭证code** ，并回传到开发者服务器。
2. 调用 [auth.code2Session](https://developers.weixin.qq.com/miniprogram/dev/api-backend/open-api/login/auth.code2Session.html) 接口，换取 **用户唯一标识 OpenID** 和 **会话密钥 session\_key**。

之后开发者服务器可以根据用户标识来生成自定义登录态，用于后续业务逻辑中前后端交互时识别用户身份。

**注意：**

1. 会话密钥 session\_key 是对用户数据进行 [加密签名](https://developers.weixin.qq.com/miniprogram/dev/framework/open-ability/signature.html) 的密钥。为了应用自身的数据安全，开发者服务器**不应该把会话密钥下发到小程序，也不应该对外提供这个密钥**。
2. 临时登录凭证 code 只能使用一次
3. 小程序注意点
   1. 如果是嵌套循环，很容易出现多个list和index，例如表视图一样，所以在小程序中可以重命名 list 和index 方法为：wx:for-index='重命名' wx:for-list="重命名"
   2. 微信小程序fiexd 定位下 scroll-view 滚动失效
      1. 解决方法: scroll-view外层容器宽度设置 为100% 就可以解决问题;
   3. wx:if 是遇 true 显示，hidden 是遇 false 显示。
   4. wx:if 在隐藏的时候不渲染，而 hidden 在隐藏时仍然渲染，只是不呈现。