**微信小程序**

# app. Json

// 小程序根目录下的 app.json 文件用来对微信小程序进行全局配置，决定页面文件的路径、窗口表现、设置网络超时时间、设置多 tab 等。

## .json文件内不可有注释

## 配置项

{

### 1.2.1 pages

"pages":[ // 所有的页面及页面地址

"pages/index/index",

"pages/logs/logs"

],

### 1.2.2 window

"window":{ // 窗口对象

"backgroundTextStyle":"light", // 下拉 loading 的样式，仅支持 dark / light 搭配enablePullDownRefresh = true使用

"enablePullDownRefresh ": true // 是否开启全局的下拉刷新。

"navigationBarBackgroundColor": "#fff", // 导航栏背景色

"navigationBarTitleText": "WeChat", // 导航栏文字

"navigationBarTextStyle":"black" // 导航栏文字色

“navigationStyle”: “default” // 导航栏样式，仅支持以下值：default 默认样式custom 自定义导航栏，只保留右上角胶囊按钮。

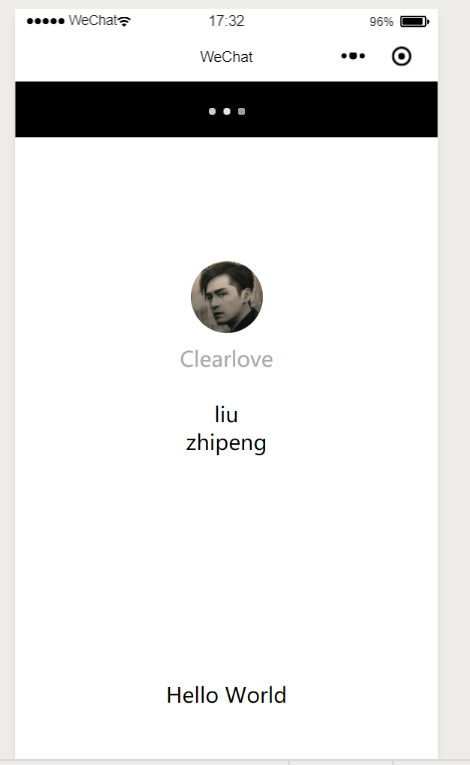
“backgroundColor”: “#fff” // 窗口的背景色

“backgroundColorBottom”: “#fff” // 底部窗口的背景色，仅 iOS 支持

“pageOrientation”: “auto” // 屏幕旋转设置，支持 auto/ portrait / landscape

}，

图片示例：

### 1.2.3 tabBar

"tabBar":{ // 如果小程序是一个多 tab 应用（客户端窗口的底部或顶部有 tab 栏可以切换页面），可以通过 tabBar 配置项指定 tab 栏的表现，以及 tab 切换时显示的对应页面。

"color": "#333", // tab 栏字体颜色

"selectedColor": "#fff", // 选中时颜色

"backgroundColor": "#eee", // 背景色

"borderStyle": "black", // 边框 仅支持 black / white

"position": "top", // tabBar 的位置，仅支持 bottom / top

"list": [ // tab 的列表，最少 2 个、最多 5 个 tab

{

"pagePath": "pages/index/index", // 跳转路径

"text": "首页", // tab文字

" iconPath ": "" // 非必填 icon路径

"selectedIconPath": "" // 非必填 选中时icon路径

},

{

"pagePath": "pages/logs/logs",

"text": "日志"

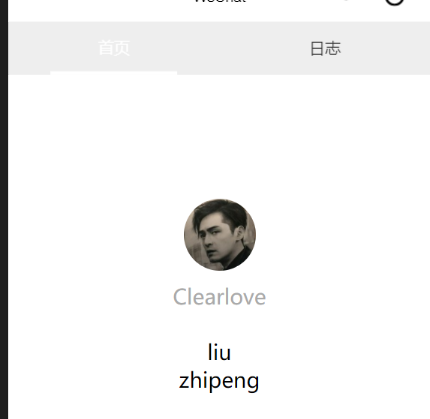
}

],

"custom": false // 自定义tabBar

}

图片示例：



### 1.2.4 networkTimeout

"networkTimeout": { // 各类网络请求的超时时间，单位均为毫秒。

"request": 10000 // 非必填 wx.request 的超时时间，单位：毫秒。

"connectSocket": 10000 // 非必填 wx.connectSocket 的超时时间 单位：毫秒。

"uploadFile": 10000 // 非必填 wx.uploadFile 的超时时间，单位：毫秒。

"downloadFile": 10000 // 非必填 wx. downloadFile的超时时间，单位：毫秒。

}

回调思想，不知接口何时返回数据，可用调用

### 1.2.5 debug

“debug”： true // 可以在开发者工具中开启 debug 模式，在开发者工具的控制台面板，调试信息以 info 的形式给出，其信息有 Page 的注册，页面路由，数据更新，事件触发等。可以帮助开发者快速定位一些常见的问题。

# page. Json

每一个小程序页面也可以使用同名 .json 文件来对本页面的窗口表现进行配置，页面中配置项会覆盖 app.json 的 window 中相同的配置项。

以下列出page.json独有的

{

"disableScroll": false // 设置为 true 则页面整体不能上下滚动。

}

# project.config. json

# sitemap. json

小程序根目录下的 sitemap.json 文件用于配置小程序及其页面是否允许被微信索引，文件内容为一个 JSON 对象，如果没有 sitemap.json ，则默认为所有页面都允许被索引；sitemap.json 有以下属性。当开发者允许微信索引时，微信会通过爬虫的形式，为小程序的页面内容建立索引。若小程序中存在不适合展示信息如用户个人信息、商业秘密、成人相关等内容，不想被微信索引到。可以通过以下方式关闭页面收录。

{

"rules": [

{

"action": "allow", // 命中该规则的页面是否能被索引，取值 "allow"、"disallow"

"params": [], // 当 page 字段指定的页面在被本规则匹配时可能使用的页面参数名称的列表

"matching": "exact", // 见下

"priority": 1, // 优先级，值越大则规则越早被匹配，否则默认从上到下匹配

"page": "pages/index/index" // 取值 "\*"、页面的路径，"\*"表示所有路径

},

{

"action": "disallow",

"page": "pages/logs/logs"

}

]

}

Matching说明：

exect 当小程序页面的参数列表等于 params 时，规则命中

inclusive 当小程序页面的参数列表包含 params 时，规则命中

exclusive 当小程序页面的参数列表与 params 交集为空时，规则命中

partial 当小程序页面的参数列表与 params 交集不为空时，规则命中

# .wxml

## 条件逻辑

使用 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>

## 列表渲染

使用 wx:for-item 指定数组当前元素的变量名，使用 wx:for-index 指定数组当前下标的变量名，wx:key 来指定列表中项目的唯一的标识符。

<view wx:for="{{array}}" wx:for-index="idx" wx:for-item="itemName" wx:key=”idx>

{{idx}}: {{itemName.message}}

</view>

wx:key 的值以两种形式提供：

1. 字符串，代表在 for 循环的 array 中 item 的某个 property，该 property 的值需要是列表中唯一的字符串或数字，且不能动态改变。
2. 保留关键字 this 代表在 for 循环中的 item 本身，这种表示需要 item 本身是一个唯一的字符串或者数字，如：

## 模板

1. <template name="odd">
2. <view> odd </view>
3. </template>
4. <template name="even">
5. <view> even </view>
6. </template>
7. <block wx:for="{{[1, 2, 3, 4, 5]}}">
8. <template is="{{item % 2 == 0 ? 'even' : 'odd'}}"/>
9. </block>

## 5.4 引用

WXML 提供两种文件引用方式import和include。

1. import 可以在该文件中使用目标文件定义的 template，如：

<import src="item.wxml"/>

<template is="item" data="{{text: 'forbar'}}"/>

2. include 可以将目标文件中除了 <template/> <wxs/> 外的整个代码引入，相当于是拷贝到 include 位置，

<include src="header.wxml"/>

<view> body </view>

<include src="footer.wxml"/>

## 5.5 共同属性

| **属性名** | **类型** | **描述** | **注解** |
| --- | --- | --- | --- |
| id | String | 组件的唯一标识 | 整个页面唯一 |
| class | String | 组件的样式类 | 在对应的 WXSS 中定义的样式类 |
| style | String | 组件的内联样式 | 可以动态设置的内联样式 |
| hidden | Boolean | 组件是否显示 | 所有组件默认显示 |
| data-\* | Any | 自定义属性 | 组件上触发的事件时，会发送给事件处理函数 |
| bind\*/catch\* | EventHandler | 组件的事件 |  |

# 6 . 场景值

场景值用来描述用户进入小程序的路径。完整场景值的含义请查看[场景值列表](https://developers.weixin.qq.com/miniprogram/dev/reference/scene-list.html)。

由于Android系统限制，目前还无法获取到按 Home 键退出到桌面，然后从桌面再次进小程序的场景值，对于这种情况，会保留上一次的场景值。

开发者可以通过下列方式获取场景值：

* 对于小程序，可以在 App 的 onLaunch 和 onShow，或[wx.getLaunchOptionsSync](https://developers.weixin.qq.com/miniprogram/dev/api/base/app/life-cycle/wx.getLaunchOptionsSync.html) 中获取上述场景值。
* 对于小游戏，可以在 [wx.getLaunchOptionsSync](https://developers.weixin.qq.com/miniprogram/dev/api/base/app/life-cycle/wx.getLaunchOptionsSync.html) 和 [wx.onShow](https://developers.weixin.qq.com/miniprogram/dev/framework/app-service/(wx.onShow)) 中获取上述场景值

# 7. app.js

App({

onLaunch: function (Object) { // 生命周期回调——监听小程序初始化。全局只触发一次。参数也可以使用 wx.getLaunchOptionsSync 获取。

}，

onShow (options) { // 生命周期回调——监听小程序启动或切前台。也可以使用 wx.onAppShow 绑定监听。

},

onHide () { // 生命周期回调——监听小程序切后台。也可以使用 wx.onAppHide 绑定监听。

// Do something when hide.

},

onError (msg) {//小程序发生脚本错误或AP调用报错时触发。也可以使用wx.onError绑定监听。 console.log(msg)

},

onPageNotFound () { // 页面不存在监听函数。也可以使用 wx.onPageNotFound 绑定监听。

wx.redirectTo({

url: 'pages/...'

}) // 如果是 tabbar 页面，请使用 wx.switchTab

},

globalData: { 开发者可以添加任意的函数或数据变量到 Object 参数中，用 this 可以访问

userInfo: null

}

})

获取到小程序全局唯一的 App 实例。

const app = getApp()

* 不要在定义于 App() 内的函数中，或调用 App 前调用 getApp() ，使用 this 就可以拿到 app 实例。
* 通过 getApp() 获取实例之后，不要私自调用生命周期函数。

# 8. page.js

//index.js

//获取应用实例

const app = getApp()

Page({

data: { // 页面的初始数据

motto: 'Hello World',

},

onLoad: function () { // 生命周期回调—监听页面加载

},

onShow: function() { // 生命周期回调—监听页面显示

// 页面出现在前台时执行

},

onReady: function() { // 生命周期回调—监听页面初次渲染完成

// 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。注意：对界面内容进行设置的 API 如wx.setNavigationBarTitle，请在onReady之后进行。

},

onHide: function() { // 生命周期回调—监听页面隐藏

// 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。

},

onUnload: function() { // 生命周期回调—监听页面卸载

// 页面卸载时触发。如wx.redirectTo或wx.navigateBack到其他页面时。同一个tabbar下的是不会卸载的

},

onPullDownRefresh: function() { // 监听用户下拉动作

// 监听用户下拉刷新事件。

需要在app.json的window选项中或页面配置中开启enablePullDownRefresh。

可以通过wx.startPullDownRefresh触发下拉刷新，调用后触发下拉刷新动画，效果与用户手动下拉刷新一致。

当处理完数据刷新后，wx.stopPullDownRefresh可以停止当前页面的下拉刷新。

},

onReachBottom: function() { // 页面上拉触底事件的处理函数

// 可以在app.json的window选项中或页面配置中设置触发距离onReachBottomDistance。

在触发距离内滑动期间，本事件只会被触发一次。

},

onShareAppMessage: function (res) { // 用户点击右上角转发

// 监听用户点击页面内转发按钮（button 组件 open-type="share"）或右上角菜单“转发”按钮的行为，并自定义转发内容。注意：只有定义了此事件处理函数，右上角菜单才会显示“转发”按钮

此事件处理函数需要 return 一个 Object，用于自定义转发内容，返回内容如下：

if (res.from === 'button') {

// 来自页面内转发按钮

}

return {

title: '日志',

path: '/pages/logs/logs',

imageUrl: "/utils/微信图片\_20191231153729.jpg"

}

},

onPageScroll: function() { // 页面滚动触发事件的处理函数

// 注意：请只在需要的时候才在 page 中定义此方法，不要定义空方法。以减少不必要的事件派发对渲染层-逻辑层通信的影响。 注意：请避免在 onPageScroll 中过于频繁的执行 setData 等引起逻辑层-渲染层通信的操作。尤其是每次传输大量数据，会影响通信耗时。

},

onResize: function() { // 页面尺寸改变时触发

// 页面尺寸变化时执行 小程序屏幕旋转时触发。

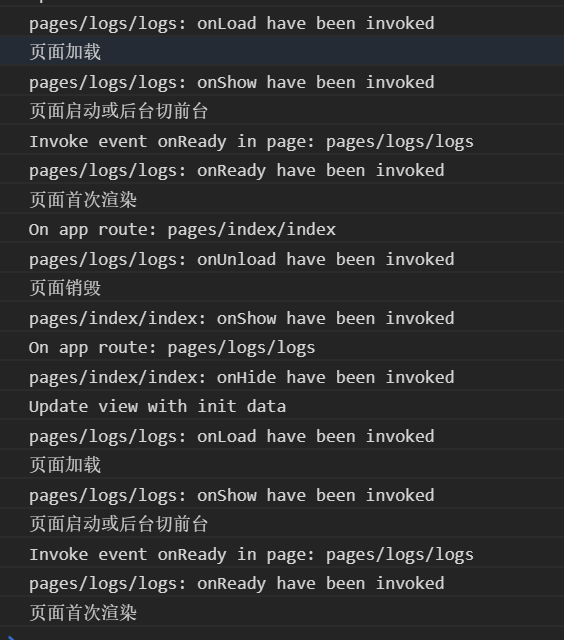
},

onTabItemTap: function(item) { // 当前是 tab 页时，点击 tab 时触发

}

})

结果：



**Page.route**

到当前页面的路径 console.log(this.route)

**Page.** **prototype.setData(Object data, Function callback)**

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）。

**注意：**

1. **直接修改 this.data 而不调用 this.setData 是无法改变页面的状态的，还会造成数据不一致**。
2. 仅支持设置可 JSON 化的数据。
3. 单次设置的数据不能超过1024kB，请尽量避免一次设置过多的数据。
4. 请不要把 data 中任何一项的 value 设为 undefined ，否则这一项将不被设置并可能遗留一些潜在问题。

**在页面中使用 behaviors**

页面可以引用 behaviors 。 behaviors 可以用来让多个页面有相同的数据字段和方法。

// my-behavior.js

module.exports = Behavior({

data: {

sharedText: 'This is a piece of data shared between pages.'

},

methods: {

sharedMethod: function() {

this.data.sharedText === 'This is a piece of data shared between pages.'

}

}

})

// page-a.js

var myBehavior = require('./my-behavior.js')

Page({

behaviors: [myBehavior],

onLoad: function() {

this.data.sharedText === 'This is a piece of data shared between pages.'

}

})

**使用 Component 构造器构造页面**

Page 构造器适用于简单的页面。但对于复杂的页面， Page 构造器可能并不好用。

此时，可以使用 Component 构造器来构造页面。 Component 构造器的主要区别是：方法需要放在 methods: { } 里面。详见后面章节

# 生命周期

