# **小程序的坑**

菜单图片尺寸为81

若数据无变化则不更新视图层，但是会继续往下执行。

无法使用本地图片背景图在小程序中显示，只能通过图片的方式，背景图只能使用网络图片的形式。

只用使用test包围的文字才可以查案选中。

小程序不认识“true”,布尔值必须加“{{true}}”才可识别。

可使用data-后接自定义属性名，来定义自定义属性，确定在for遍历得到的是哪一数据。该属性可在事件中的到。

因为获取是异步的，当渲染的时候会报错，加上wx:if就可以实现避免报错。在图片渲染时候报错。

非需要数据重新渲染的不用使用setData会增大渲染开销。

原生的类似video会置顶显示，所以在需要出蒙版的时候可以写点击在弹出蒙版的同时删除组件。

切图，任何图标都要是正方形，相同的要使用统一的大小正方形。

在小程序中，即时不需要写js的代码，那js文件中也必须要加上Page({})。

Img标签在更改前后两个数据相同时不可重复请求同一个路径，先置空就可正常显示。

**页面必须写在一个单独的文件夹内。**

**写标签必须一行，太多了可以挤下去但是内容跟标签必须在一行，否则高度会莫名其妙多一节。**

**在组件wxss中不应使用ID选择器、属性选择器和标签名选择器。**

**this.setData({**

**[key]:value**

**})**

**[button](https://mp.weixin.qq.com/debug/wxadoc/dev/component/button.html) 组件通过设置 open-type="contact" 亦可进入客服会话**

**App注册完成之后加载页面，但是登录事件为异步。**

**Scroll-view里面不能直接放东西，把里面的放在一个view里，相邻的也要放在一个view里面，不能散开放。**

**上拉必须有一块固定不动。**

**Image的mode属性不可多用，在初始加载时图片会变形，最好使用文字图标或者写死宽高。可设置mode截取长边留下短边，然后宽高写死。**

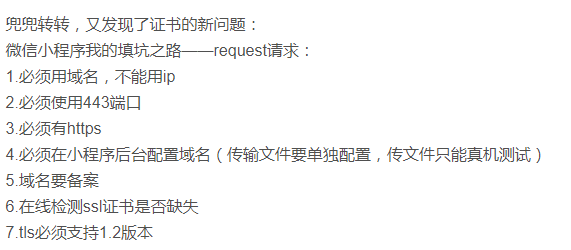
**所有原生组件层级最高，暂时无解。例：map，textarea，input，vedio。。。**

**input不可透明，没有hidden只能display:none;获取焦点只能通过点击事件去获取焦点。**

**页面app实例外的变量重新打开不会重置，切记。**

**分享事件没写则没有分享功能。**

**直接新建Page会自带分享**



# **主要文件**

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。调用框架提供的丰富的 API，如本例的同步存储及同步读取本地数据。

## app.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。

## Pages

接受一个数组，每一项都是字符串，来指定小程序由哪些页面组成。每一项代表对应页面的【路径+文件名】信息，数组的第一项代表小程序的初始页面。小程序中新增/减少页面，都需要对 pages 数组进行修改。

## window

用于设置小程序的状态栏、导航条、标题、窗口背景色。

可设置顶部是否自定义

"navigationStyle":"custom"

## 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。

## networkTimeout

可以设置各种网络请求的超时时间。

## debug

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 。 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.json文件来对本页面的窗口表现进行配置。 页面的配置比app.json全局配置简单得多，只是设置 app.json 中的 window 配置项的内容，页面中配置项会覆盖 app.json 的 window 中相同的配置项。

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。

# **结构**

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](https://mp.weixin.qq.com/debug/wxadoc/dev/framework/app-service/page.html)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.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.wxss 是页面的样式表：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。

Utils是公用的文件，可以写各种方法。

## App()

App() 函数用来注册一个小程序。接受一个 object 参数，其指定小程序的生命周期函数等。

前台、后台定义： 当用户点击左上角关闭，或者按了设备 Home 键离开微信，小程序并没有直接销毁，而是进入了后台；当再次进入微信或再次打开小

## getApp()

全局的 getApp() 函数可以用来获取到小程序实例。

App() 必须在 app.js 中注册，且不能注册多个。

不要在定义于 App() 内的函数中调用 getApp() ，使用 this 就可以拿到 app 实例。

不要在 onLaunch 的时候调用 getCurrentPages()，此时 page 还没有生成。

通过 getApp() 获取实例之后，不要私自调用生命周期函数。

## 小程序生命周期

<https://mp.weixin.qq.com/debug/wxadoc/dev/framework/app-service/page.html>

小程序的生命周期介绍。

# **标签**

view（类似div）、text（类似span）、icon等等，class同html指定样式，bindtap绑定事件（类似onclick）

block配合wx:if、wx:for使用，标明if、for的作用范围，block 只作为列表渲染用，自身不显示在页面中，不起布局作用。

1rpx = 0.5px = 1物理像素。http://blog.csdn.net/qq\_31383345/article/details/52817807

## scroll-view可滚动视图区域。

通过scroll-x与Y来控制那个方向的滚动，但是要对scroll-view设置好大小。

scroll-top与left用来设置初始化的时候滚动的位置在哪里。

Scroll中的子元素必须为行内块元素，其本身必须设置强制不换行：white-space: nowrap;

里面不可放各种原生组件与输入组件。

#### **image**

该组件很恶心，若是宽高固定就写死，如果宽固定高不固定就使用mode属性写个保持纵横比。但是应设置溢出隐藏。该组件在初始化时会保持image的默认宽高，会出现一瞬间的变形。

## Vedio

注：video 封面poster图片在真机IOS下会被撑大，不自适应。必须按照比例返回固定比例图片大小的图片，这个在不同机型上很难办到

seek(0)以后，再pause()，但是这样做会出现另外的bug：

第一次播放时，播放当前视频，其他视频会一起加载

除了第一次播放，虽然实现了伪停止效果，但是封面图会不见了

cover-image 里的图片不支持放大吗？

mp4视频用video组件播放会卡顿、绿屏,尤其是在快进、拖动时间条时.在video组件上，加 试试加个custom-cache=”{{false}}”属性，可以完美解决。

video封面: poster

使用的poster属性，开发工具上图片闪一下就没了。在video里用

cover-view和cover-image，开发工具里正常显示，但是图片不能自适应

video组件在填写poster属性后..转发可能引起小程序崩溃.

在一个页面内,放置2个以上的填写了poster属性的video组件,

之后在android端转发会引起小程序崩溃…不知道为什么…

# **脚本层**

页面管理

框架 管理了整个小程序的页面路由，可以做到页面间的无缝切换，并给以页面完整的生命周期。开发者需要做的只是将页面的数据，方法，生命周期函数注册进 框架 中，其他的一切复杂的操作都交由 框架 处理。

Page为定义页面数据区域。

Page({

})

# **Page**

Page() 函数用来注册一个页面。接受一个 object 参数，其指定页面的初始数据、生命周期函数、事件处理函数等。

触发事件更改数据

****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. ****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****。

# **Map**

注意: covers 属性即将移除，请使用 markers 替代。

display 'BYCLICK':点击显示; 'ALWAYS':常显。 只有设置后才会显示气泡。

1. tip: map 组件是由客户端创建的原生组件，它的层级是最高的。
2. tip: 请勿在 scroll-view 中使用 map 组件。
3. tip: css 动画对 map 组件无效。
4. tip: map 组件使用的经纬度是火星坐标系，调用 wx.getLocation 接口需要指定 type 为 gcj02

注：不可直接选择其他地图，但可通过微信内置地图跳转到其他地图App。

# **表单**

Form中需要加入catchsubmit="formSubmit" 提交数据 catchreset="formReset" 清空表单。

表单元素有设置好的样式可以直接套用，button有多种内置样式。

例：<button type="primary" formType="submit" loading="true">Submit</button>

# **动画**

**华而不实的东西。**

在周期函数中以实例的方法创建createAnimation对象，调用实例的方法来描述动画。最后通过动画实例的export方法导出动画数据传递给组件的animation属性。有多种动画效果。

初始化动画名

alertBox: {},

在加载完的钩子函数中设置动画，

let animation = wx.createAnimation({

duration: 500,

timingFunction: 'ease',

})

this.animation = animation;

在定义事件中执行后跟链式动画效果，最后以step结尾，表明动画完成。最后在setData中进行更新。每执行一次export则执行一次动画。

close() {

this.animation.top("100%").opacity(0).step();

this.setData({

alertBox: this.animation.export()

})

},

页面引用

<view class='b-box dis\_f fle\_f' animation="{{alertBox}}" style='top:100%;opacity:0'>

# **页面跳转**

页面内跳转靠open-type设置跳转的方式：

例：<navigator open-type="navigate" url="/pages/pay/index/view/map/map" hover-class="navigator-hover"></navigator >

可以设置点击时的效果。

Js跳转至tobar页面需要使用

wx.switchTab({

url: '/pages/success/index',

success: function(res) {},

fail: function(res) {},

complete: function(res) {},

})

跳转至普通页面可用

redirectTo或者navigateTo

navigateTo只能跳转5次，为了避免多次跳转对用户的困惑。

而redirectTo是重定向页面，无法发返回，navigateTo则可以返回。

# **获取页面路径**

var pages = getCurrentPages() //获取加载的页面

var currentPage = pages[pages.length-1] //获取当前页面的对象

var url = currentPage.route //当前页面url

var options = currentPage.options //如果要获取url中所带的参数可以查看options

原生组件中的view会覆盖在原生组件之上，vedio的pros属性又无法宽度百分百。开发工具上的显示与手机不同。

# **数据渲染**

## 组件属性(需要在双引号之内)

<view id="item-{{id}}"> </view>

## 控制属性(需要在双引号之内)判断语句

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

关键字(需要在双引号之内)

<checkbox checked="{{false}}"> </checkbox>

特别注意：不要直接写 checked="false"，其计算结果是一个字符串，转成 boolean 类型后代表真值。

运算

可以在 {{}} 内进行简单的运算。

## wx:key

如果列表中项目的位置会动态改变或者有新的项目添加到列表中，并且希望列表中的项目保持自己的特征和状态（如 <input/> 中的输入内容，<switch/> 的选中状态），需要使用 wx:key 来指定列表中项目的唯一的标识符。

wx:key 的值以两种形式提供

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。

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：

当数据改变触发渲染层重新渲染的时候，会校正带有 key 的组件，框架会确保他们被重新排序，而不是重新创建，以确保使组件保持自身的状态，并且提高列表渲染时的效率。

如不提供 wx:key，会报一个 warning， 如果明确知道该列表是静态，或者不必关注其顺序，可以选择忽略。

# **组件**

组件的wxml文件正常书写，wxss也是，js文件不同于正常文件，可以设置接受外面传入的值也可向外面传出值。传入值更改后有回调函数。

# 组件间关系

组件之间的通信可以通过Component来实现，定义相关组件的联系。

关联一类组件，若他们有一个共同的behavior，则则在 relations 关系定义中，可使用这个behavior来代替组件路径作为关联的目标节点：

# **模版**

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

例：<template name="msgItem">

<view>

<text> {{index}}: {{msg}} </text>

<text> Time: {{time}} </text>

</view>

</template>

**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'

}

}

})

is 属性可以使用 Mustache 语法，来动态决定具体需要渲染哪个模板：

<template name="odd">

<view> odd </view>

</template><template name="even">

<view> even </view>

</template>

<block wx:for="{{[1, 2, 3, 4, 5]}}">

<template is="{{item % 2 == 0 ? 'even' : 'odd'}}"/>

</block>

模板拥有自己的作用域，只能使用data传入的数据。

引入wxss使用方法为在目标wxss文件中：@import "common.wxss";

# **WXS 模块**

作为页面格式化的作用。

传送门：<https://developers.weixin.qq.com/miniprogram/dev/framework/view/wxs/>

类似于javaScript标签，不可写在wxss中，只能在wxs模块中进行写入。WXS 代码可以编写在 wxml 文件中的 <wxs> 标签内，或以 .wxs 为后缀名的文件内。每个模块都有自己独立的作用域。即在一个模块里面定义的变量与函数，默认为私有的，对其他模块不可见。

只可做对数据的更改操作后返回，无法定义事件和行为。只能对数据操作后返回数据。

一个模块要想对外暴露其内部的私有变量与函数，只能通过 module.exports 实现。

例：module.exports = {

foo: foo,

bar: bar

};

在接受页面的page外引入，但是只能用相对路径，不能用绝对路径。

例：var data=require(“../../data/post-data.js”)

## require函数

在.wxs模块中引用其他 wxs 文件模块，可以使用 require 函数。

例：  
var foo = "'hello world' from tools.wxs";var bar = function (d) {

return d;

}module.exports = {

FOO: foo

};module.exports.msg = "some msg";

// /pages/logic.wxs

var tools = require("./tools.wxs");

console.log(tools.FOO);

console.log(tools.bar("logic.wxs"));

<!-- /page/index/index.wxml -->

<wxs src="./../logic.wxs" module="logic" />

## module 属性

module 属性是当前 <wxs> 标签的模块名。在单个 wxml 文件内，建议其值唯一。有重复模块名则按照先后顺序覆盖（后者覆盖前者）。不同文件之间的 wxs 模块名不会相互覆盖。

module 属性值的命名必须符合下面两个规则：

首字符必须是：字母（a-zA-Z），下划线（\_）

剩余字符可以是：字母（a-zA-Z），下划线（\_）， 数字（0-9）

****示例代码：****

<!--wxml-->

<wxs module="foo">

var some\_msg = "hello world";

module.exports = {

msg : some\_msg,

}</wxs><view> {{foo.msg}} </view>

页面输出：

hello world

上面例子声明了一个名字为 foo 的模块，将 some\_msg 变量暴露出来，供当前页面使用。

## src 属性

src 属性可以用来引用其他的 wxs 文件模块。

引用的时候，要注意如下几点：

只能引用 .wxs 文件模块，且必须使用相对路径。

wxs 模块均为单例，wxs 模块在第一次被引用时，会自动初始化为单例对象。多个页面，多个地方，多次引用，使用的都是同一个 wxs 模块对象。

如果一个 wxs 模块在定义之后，一直没有被引用，则该模块不会被解析与运行。

<wxs src="./../comm.wxs" module="some\_comms"></wxs>

### 注意

<wxs> 模块只能在定义模块的 WXML 文件中被访问到。使用 <include> 或 <import> 时，<wxs> 模块不会被引入到对应的 WXML 文件中。

<template> 标签中，只能使用定义该 <template> 的 WXML 文件中定义的 <wxs> 模块。

# **作用域与数据传递**

# **事件**

注意：通过事件改变的内容也必须经过setdata的更改才可实现。更改列表数据需要更新一下最新的数据。

例：this.setData({

startList: this.data.startList

})

Tap事件为点击，默认加上bind即为事件全名，

前面加上catch的为非冒泡事件。

# **页面间传输数据**

1，url拼接的方式

<navigator url="/pages/mode/mode?name='{{user[0].name}}'&age={{user[0].age}}">点击跳转</navigator>

或

wx: wx.navigateTo({

url: '/pages/mode/mode?name=ni'

})

只可以传输字符串，或者转换成字符串。url后拼接?id(参数名字)=要传递的值 (如果多个参数用&分开 &name=value&…….)

获取的时候在跳转后的页面中的钩子函数中进行获取。

onLoad: function (options) {

console.log(options) 键值对的形式。

},

2，Js获取页面结构树，获取上一个页面实例，更改值并返回页面。

例：const pages = getCurrentPages();

const prePage=pages[pages.length-2];

prePage.setData({

taps: e.target.id

})

wx.navigateBack({

delta: 1

})

# **模版**

在要使用组件的页面的wxml文件中引用组件，同时使用模版，传入担任“命名空间”的对象

<import src="../../component/login-pannel/login-pannel.wxml"/> 使用include会直接把模版代码复制过来，适用于模版嵌套模版的情况。使用import引入的模版无法使用模版内引用的模版。

<template is="loginPannel" data="{{ ...\_\_lgpanel\_\_ }}"/> 传参给组件的方式是使用data传入

如果你的组件需要在多个页面中使用，那么在app.wxss中引入一次组件样式即可；如果只需要在摸个特定的页面使用，只需在该页面的样式文件中引入组件样式。

@import "./component/login-pannel/login-pannel.wxss";

如果你的组件需要在多个页面中使用，你需要在app.js中引用，同时传给App()：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | import { LoginPannel } from './component/login-pannel/login-pannel'  App({  LoginPannel,  \_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 () {  wx.getUserInfo({  success: function (res) {  that.globalData.userInfo = res.userInfo  typeof cb == "function" &&  cb(that.globalData.userInfo)  }  })  }  })  }  }  }) |

然后，在使用的页面的生命周期函数onLoad中“注册”该组件：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | onLoad: function () {  // 调用应用实例的方法获取全局数据  let app = getApp()    // “注册”组件  new app.LoginPannel()    app.\_getUserInfo((userInfo) => {  //更新数据  this.setData({  userInfo:userInfo  })  })  } |

然后，在这个页面中就可以愉快的使用组件啦：

**注意：模版的样式最好放在全局，无法使用事件，只可展示和使用跳转。**

# **小程序使用矢量图**

将其下载到本地，拉出来后缀是ttf的文件，在<https://transfonter.org>转成base64的格式（选择base64开启，其他的都关了）转换后下载下来并且将转换文件中的css文件中的内容都复制到新的wxss文件。之前的那个font—css文件除了引入那一段其他的都复制到转化后的文件内后缀为wxss，在需要的wxss中引入即可了。

# **富文本**

使用wxParse插件，

仓库地址：<http://blog.csdn.net/zhuming3834/article/details/74280038>

# **图片预览功能**

地址：[HTTP://mp.weixin.qq.com/debug/wxadoc/dev/api/media-picture.html#wxpreviewimageobject](HTTP://mp.weixin.qq.com/debug/wxadoc/dev/api/media-picture.html" \l "wxpreviewimageobject)

//图片预览

wx.previewImage({

current: src, // 当前显示图片的http链接

urls: imgList // 需要预览的图片http链接列表

})

# **交互反馈**

各种弹窗信息。

地址：<HTTP://mp.weixin.qq.com/debug/wxadoc/dev/api/api-react.html>

注：其中的颜色不可使用英文单词，ios上不显示。

# **下拉加载上拉刷新**

在app.json中定义enablePullDownRefresh，则全都开启，在单独页面定义则单独开启。



下拉样式

例：enablePullDownRefresh:true

传送门：<https://mp.weixin.qq.com/debug/wxadoc/dev/api/pulldown.html#onpulldownrefresh>

# **转发**

## onShareAppMessage(options)可在page内定义转发，也可在页面内点击转发，通过给 button 组件设置属性 open-type="share"

# **小程序之间的跳转**

<https://mp.weixin.qq.com/debug/wxadoc/dev/api/navigateToMiniProgram.html>

wx.navigateToMiniProgram(OBJECT)，在小程序之间跳转的前提是两个小程序都是在同一个公众号下边。

# **小程序推送信息**

<https://mp.weixin.qq.com/debug/wxadoc/dev/api/notice.html>

小程序中的模版信息，可以推送信息，仅当用户使用小程序发生过提交表单或者下单后才可推送信息，推送的信息条数跟下单和提交表单的次数相同。

# **客服信息**

可用按钮contact-button但最好别用或者使用button的open-type指定为客服按钮。客服按钮样式不可改，但可改button的样式。

例：<button open-type="contact" session-from="weapp">

<image src="/pages/images/m\_service\_ico.png" mode="widthFix"></image>客服

</button>

点击进入客服会话可通过session-from传值指定回话来源。

<https://mp.weixin.qq.com/debug/wxadoc/dev/api/custommsg/receive.html>

# **上传图片或文件**

**传送门：**

**<https://mp.weixin.qq.com/debug/wxadoc/dev/api/network-file.html#wxuploadfileobject>**

filePath只能为string ，一次只能上传一个，

**uploadTask**

可以监听上传的进度，已经上传的长度，预计上传时间，或者取消上传。

# **分包加载**

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

在构建小程序分包项目时，构建会输出一个或多个功能的分包，其中每个分包小程序必定含有一个****主包****，所谓的主包，即放置默认启动页面/TabBar 页面，以及一些所有分包都需用到公共资源/JS 脚本，而****分包****则是根据开发者的配置进行划分。

公用页面的文件和js都放在分包内，不占用空间。

# **扫描二维码进入页面**

获取二维码中获取的值在query中

