# 微信小程序

### 推荐长度单位rpx:

Rpx与px的转化： 【屏幕宽度/750】

Iphone6: 375 \* 667 1rpx = 0.5px = 1物理像素（375/750 = 0.5px）

750rpx = 100%屏幕宽度

375rpx = 50%屏幕宽度

Touch: 375 = 750rpx

[ 标签 1](#_Toc1460362)

[Navigator(页面链接) 1](#_Toc1460363)

[ 方法（函数） 2](#_Toc1460364)

[Bindtap与catchtap 2](#_Toc1460365)

[switchTab、navigateTo、navigateBack与redirectTo 2](#_Toc1460366)

[tabbar样式： 3](#_Toc1460367)

[setStorage 与 setStorageSync 3](#_Toc1460368)

[showToast、showLoading、hideLoading、showActionSheet(消息弹框) 4](#_Toc1460369)

[showToast与hideToast(隐藏) 4](#_Toc1460370)

[showLoading与hideLoading(隐藏) 5](#_Toc1460371)

[showActionSheet 5](#_Toc1460372)

[showNavigationBarLoading、hideNavigationBarLoading、setNavigationBarTitle、setNavigationBarColor 5](#_Toc1460373)

[BackgroundTextStyle、backgroundColor 6](#_Toc1460374)

[导航栏tabbar 6](#_Toc1460375)

Username：startAppTest

Id：gh\_2cc9b5e82743

AppID：wx948ae84e43c5b069

## 标签

#### Navigator(页面链接)

url：目标页面路径

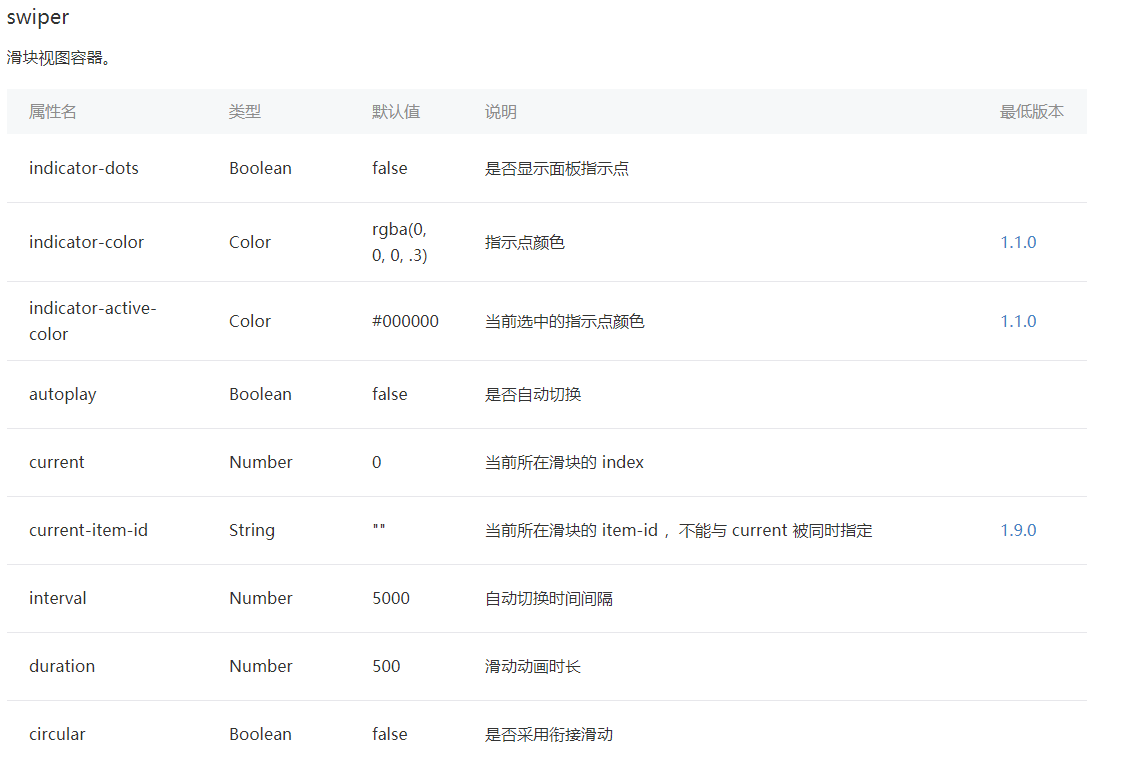
open-type：navigate/ redirect/ switchTab/ relaunch/ navigateBack (open-type=’navigateBack与wx.navigateBack类似，前者返回上一级,后者根据delta层数返回对应页面’)

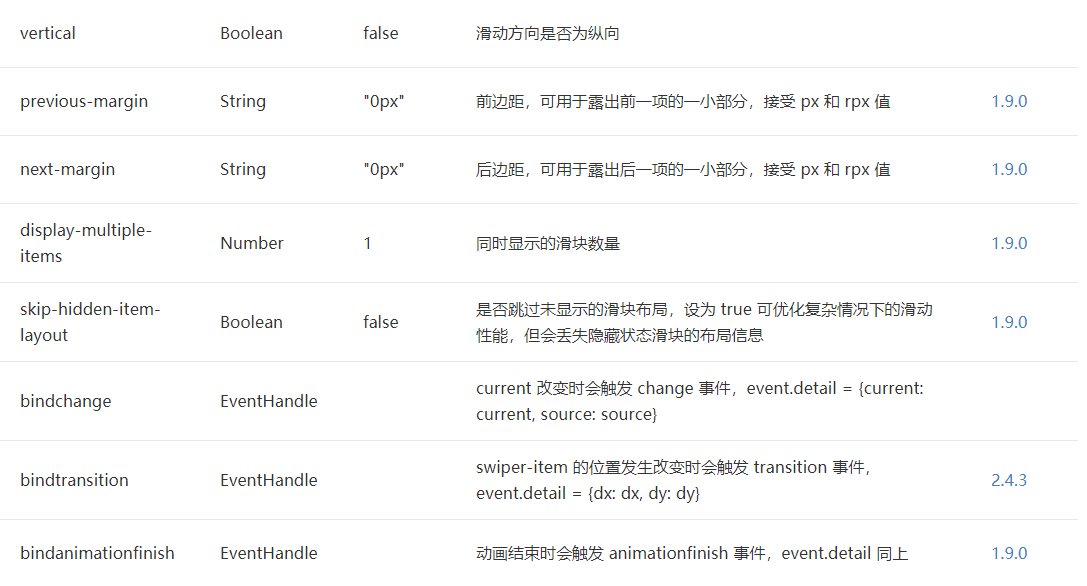
#### progress(进度条)

<progress class=’xx’ precent=’20’ show-info></progress>



#### Swiper(滑块视图容器)





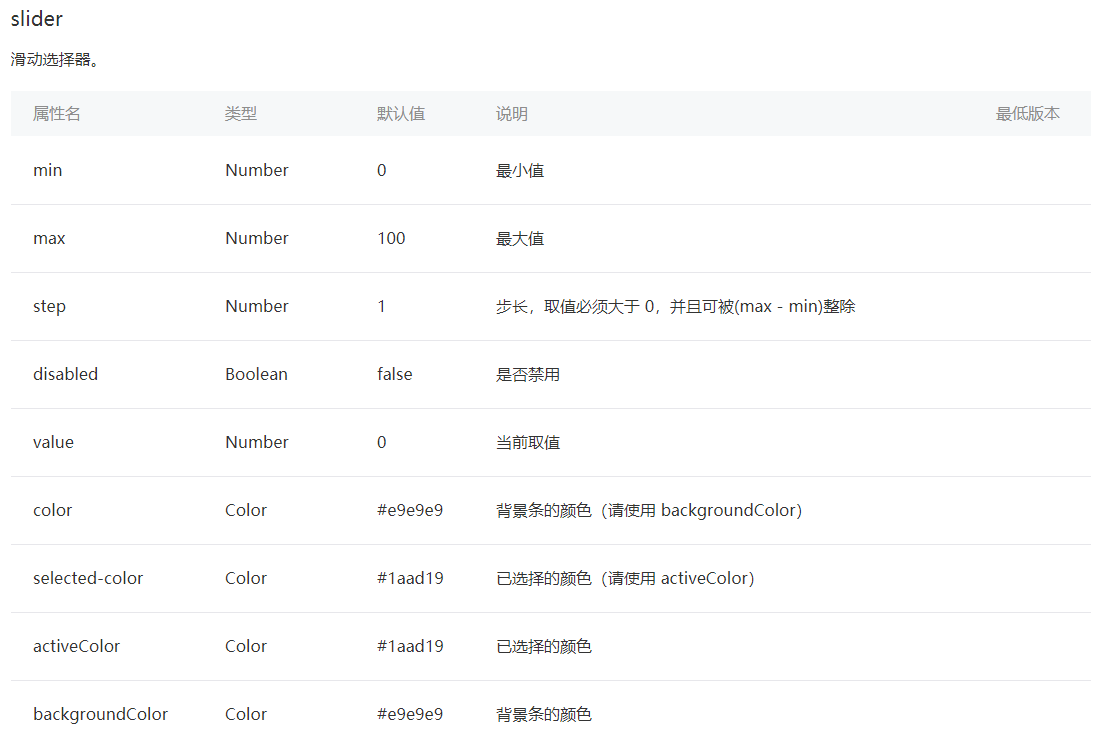
#### Block(类似继承)

在block标签中使用wx:for={{content}}，遍历数组并获取数据，在block内部的元素可以使用item获取content,此时的item = block中for的数据，类似于子元素继承了block for内容，索引为item



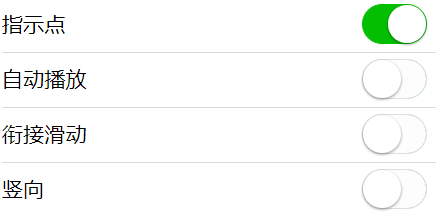
#### Slider(滑动选择器)







#### Switch(按钮选择器)



#### picker

使用方法：

data: address:[‘第一’,’第二’,’刮大风’]

<picker mode='selector' range="{{address}}" value="labelIndex" bindchange='pickerValue'>

<label>选择</label>

<text>{{address[labelIndex]}}</text>

</picker>

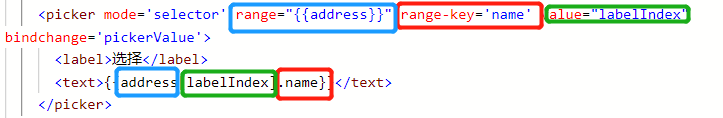
range：显示列表的数据

value：下标值（自动），如上，labelIndex：在此随意定义的字段，picker自动将数组与此

下标匹配



data: address:[{name:’第一’},{name:’第二’},{name:’刮大风’}]



#### map

只有cover-\*可覆盖于map之上，如cover-view、cover-image

## 方法（函数）

#### **Bindtap与catchtap**

bindtap绑定的事件会冒泡向外层悬浮，catchtap只对绑定的元素作用（类似于js的event=> target 与 currentTarget,target返回触发事件的元素，currentTarget返回绑定事件的元素）。



#### switchTab、navigateTo、navigateBack与redirectTo

redirectTo：关闭当前页（卸载），跳转到指定页

navigateBack：根据页面栈返回指定页面 （wx.navigateBack({ delta: 2 })）,与navigateTo关联

navigateTo：保留当前页（隐藏），并将此页面存入页面栈，跳转到指定页

wx.navigateTo/redirectTo({ url: '../logs/logs' }) //跳转的页面路径

switchTab：只能运用于跳转tabbar页面，并关闭其他非tabbar页面(类似z-index:9999效果)，tabbar之间做切换【其他页面无法跳转到tabbar页面】，

switchTab与app.json中tabBar关联：



wx.switchTab({ url: ‘./index’}) // tabbar的显示页面的路径样式

###### tabbar样式：





Position为top时，图标隐藏：





#### setStorage 与 setStorageSync

区别：setStorage为异步存储数据，setStorageSync为同步存储数据，异步存储（不论是否存储成功都执行下面程序（方便），同步存储失败停止运行后面程序（更安全））

//异步设置storage

wx.setStorage({

key: 'key',

data: 'value'

})

//同步设置storage

wx.setStorageSync('key', 'value')

//获取storage

wx.getStorage({

key: 'key',

success:function(res){

console.log(res.data);

}

})

//全部删除storage

wx.clearStorage();

wx.clearStorageSync();

//删除指定key storage

wx.removeStorage({

key: 'key',

success:function(res){

console.log(res.data);

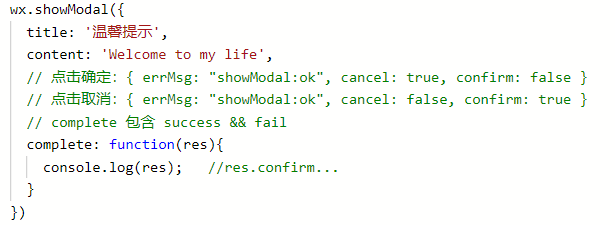
}

})

wx.removeStorageSync('key')

#### showToast、showModal、showLoading、hideLoading、showActionSheet(消息弹框)

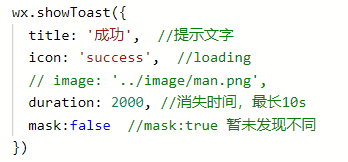
###### showModal





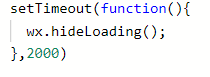
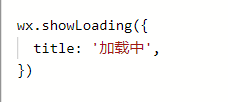
###### showToast与hideToast(隐藏)

若需要使用自定义图片展示，可用image:src属性，image优先于icon

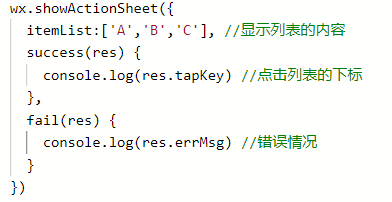


###### showLoading与hideLoading(隐藏)

与showToast=>icon: ‘loading’类似，但showLoading不会主动消失，需主动隐藏



###### showActionSheet



#### 【下拉刷新】

**showNavigationBarLoading、hideNavigationBarLoading、**

**setNavigationBarTitle、setNavigationBarColor**

首先在app.json中设置允许下拉刷新：

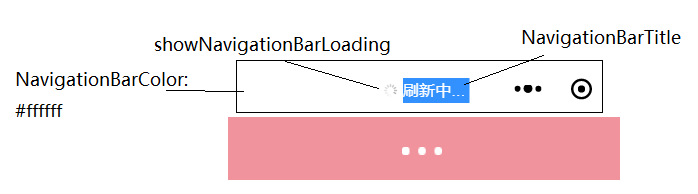
enablePullDownRefresh: true,再在需要的页面的js中设置以后方法：

【

局部页面禁止下拉刷新：在对应页面的json中设置"enablePullDownRefresh": false

】

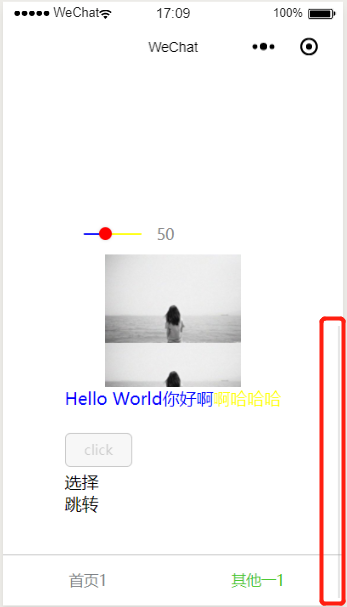
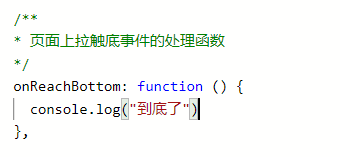




#### 【上拉加载】

**onReachBottom：page滚动到底部时触发(滚动条至底部)**

**bindscrolltolower： scroll-view(滚动到底部/右边时触发)**

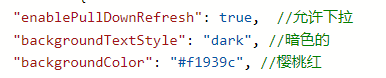
 

区别：onReachBottom范围为整个page页面，当page内容高度超过屏幕出现滚动条并滚

动至底部时触发

bindscrolltolower为scroll-view专属的一个事件，其滚动至底部触发此事件

#### BackgroundTextStyle、backgroundColor

#### 导航栏tabbar



**自定义导航：**

**app.json 🡪**

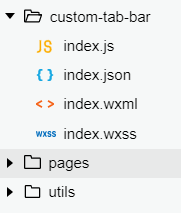
**1、添加**

**"usingComponents": {**

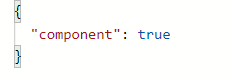
**"custom-tab-bar": "/custom-tab-bar/index"**

**}**

2、创建component



3、component.js 设置component



4、导入组件

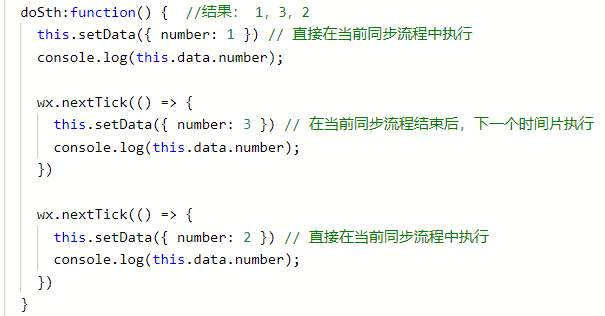


要点提示：app.json中的pages数值中的第一个数组需要在tabbar中，否则tabbar导航栏不会显示，且pages中的第一个页面配置会成为程序首页，tabbar中的显示跳转wx.switchTab，当组件中的path与app.json中设置的tabbar相同时，组件的path被覆盖

【wx.switchTab({url:'./popUpWindow'})】

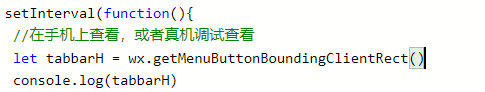
#### nextTick

延迟执行，类似于线程优先级，设置nextTick的内容在同一环境中最后执行，多个nextTick按照排列先后顺序执行。



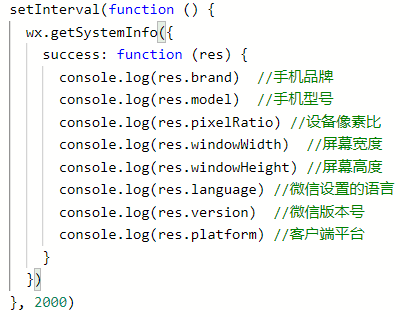
#### getMenuButtonBoundingRect

获取右上角导航栏中按钮位置信息：left,top,width,height…



#### getSystemInfo

获取系统信息





#### Request



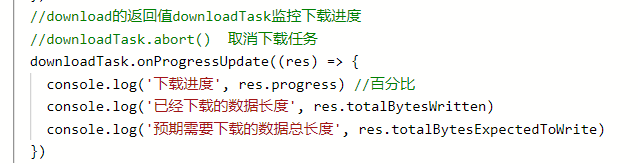
特别提示：当http访问本地文件时，method使用GET方式，https对应POST

#### downloadFile（下载、保存文件）

wx.downloadFile 下载文件默认地址：【C:\Users\刘盈\AppData\Local\微信web开发者工具\UserData\WeappFileSystem\o6zAJs8bspIiNmygwtsndbYO\_e\_w\wx948ae84e43c5b069\tmp】

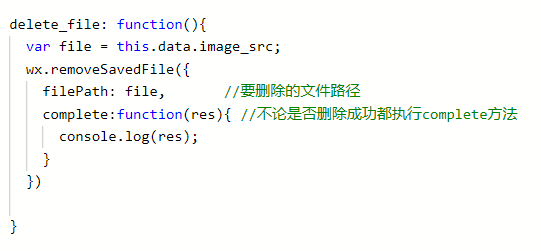
wx.saveFile保存文件默认地址：【C:\Users\刘盈\AppData\Local\微信web开发者工具\User Data\WeappFileSystem\o6zAJs8bspIiNmygwtsndbYO\_e\_w\wx948ae84e43c5b069\store】





要点提示：预览文件时谨慎使用**tempFilePath**，文件加载完成后tempFilePath为文件临时保存地址，将临时文件移动到本地保存后，原先临时文件消失，使用之前的临时地址预览会产生404（找不到文件）错误。

#### removeSavedFile(删除缓存文件)



系统自动识别选择的文件是否缓存文件

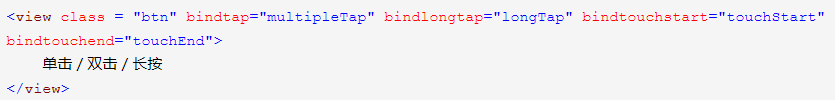
#### hidden、wx:if、display:none/block

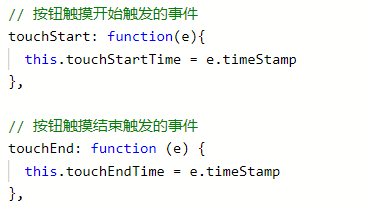
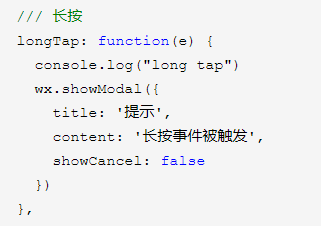
hidden:隐藏元素【hidden: ‘true’】{当此元素display不为block时，hidden无效}

wx:if 隐藏元素【wx:if=’true’】

区别：wx:if为false时，不渲染元素，wx:if变为true时，重新渲染元素，hidden渲染元素，只是控制是否显示。当操作频繁时不建议使用wx:if，消耗性能

#### 单击、双击、长按事件





#### 数组

###### 修改数组内容

使用javascript的splice()：【array.splice(“data”,start\_index,end\_start)】

由于小程序中this.data.content = xxx只能在js内部修改，想要在html中同步更新内容使用setData({content: this.data.xxx}),因此可以使用this.data.content的方式让替换、修改数组内容，再setData给数组【content: this.data.content】，避免setData中不便于操作数组的情况

###### 查询、删除、替换数组内容

Javascript => indexOf(“query\_content”),查找到内容返回内容下标，找不到返回-1

Javascript => splice(start\_index,end\_index,”updata\_content”),当updata\_content不存在时，执行删除操作，其存在时，使用其内容替换应被删除的内容

###### 添加数组内容

Javascript => push/pop 从数组尾端（添加/删除）数组内容

Javascript => unshift/shift 从数组首端（添加/删除）数组内容

## 组件（Component）

### 生命周期

created:function(){

// 组件生命周期函数，在组件实例进入页面节点树时执行， 注意此时不能调setData

console.log('Component-1 >> created');

},

attached:function(){

// 组件生命周期函数，在组件实例进入页面节点树时执行。

console.log('Component-1 >> attached');

},

ready:function(){

// 在组件布局完成后执行，此时可以获取节点信息

console.log('Component-1 >> ready');

},

moved:function(){

// 在组件实例被移动到节点树另一个位置时执行

console.log('Component-1 >> moved');

},

detached:function(){

// 在组件实例被从页面节点树移除时执行

console.log('Component-1 >> detached');

},

lifetimes:{

// 组件生命周期声明对象，将组件的生命周期收归到该字段进行声明，原有声明方式仍旧有效，如同时存在两种声明方式，则lifetimes字段内声明方式优先级最高

created:function(){

console.log('Component-1 lifetimes >> created');

},

attached:function(){

console.log('Component-1 lifetimes >> attached');

},

ready:function(){

console.log('Component-1 lifetimes >> ready');

},

moved:function(){

console.log('Component-1 lifetimes >> moved');

},

detached:function(){

console.log('Component-1 lifetimes >> detached');

}

},

pageLifetimes:{

// 组件所在页面的生命周期声明对象，目前仅支持页面的show和hide两个生命周期

show:function(){

console.log('Component-1 pageLifetimes >> Show');

},

hide:function(){

console.log('Component-1 pageLifetimes >> Hide');

}

}

## 页面（Page）

### 生命周期

Page({

data:{

},

onLoad:function(options){

//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路径中的参数。

//参数 options Object 打开当前页面路径中的参数

console.log('pageA >> onLoad , options ::',options);

},

onReady:function(){

//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。

console.log('pageA >> onReady');

},

onShow:function(){

//页面显示/切入前台时触发

console.log('pageA >> onShow');

},

onHide:function(){

//页面隐藏/切入后台时触发。

console.log('pageA >> onHide');

},

onUnload:function(){

//页面卸载时触发。

console.log('pageA >> onUnload');

},

// 自定义方法

navigateToC:function(){

//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 可以返回到原页面。

console.log('%cpageA====navigateToC=====','color:red');

wx.navigateTo({

url:'/pages/page-c/index'

});

},

redirectToC:function(){

//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

console.log('%cpageA=====redirectToC======','color:red');

wx.redirectTo({

url:'/pages/page-c/index'

});

},

reLaunchToC:function(){

//关闭所有页面，打开到应用内的某个页面

console.log('%cpageA====reLaunchToC====','color:red');

wx.reLaunch({

url:'/pages/page-c/index'

})

}

});

## mpvue

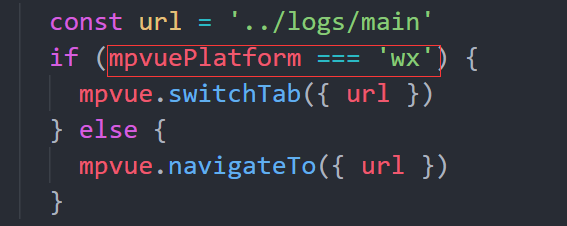
1. 创建页面

新建文件夹，内部创建index.vue和main.js文件（可复制其他页面）

重启微信开发者工具，导入项目

1. 页面跳转

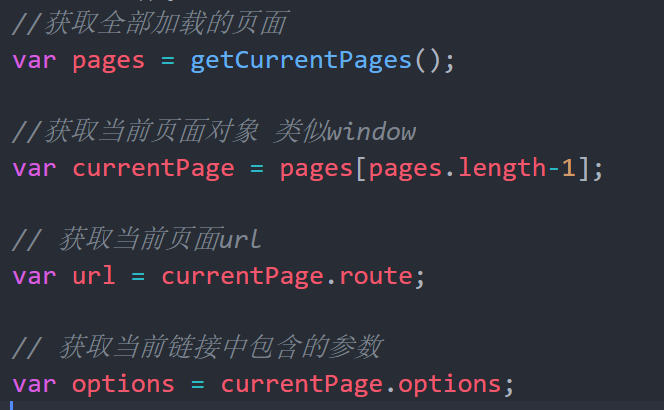
在app.json中pages | tabBar添加目标页面



传值：



获取数据：



v-bind:href 字符部分用’ ’裹住

## 小程序（App）

//app.js

App({

onLaunch: function (options) {

// 小程序初始化完成时触发，全局只触发一次。

// options说明：

// path:打开小程序的路径

// query 打开小程序的query

// scene 打开小程序的场景值

// shareTicket 转发信息相关

// referrerInfo 当场景为由另一个小程序或公众号或App打开时，返回此字段

console.log('app >> onLaunch ， options::',options);

},

onShow: function(options){

// 小程序启动，或从后台进入前台显示时触发

// 参数与onLaunch一致

console.log('app >> onShow， options :: ',options);

},

onHide:function(){

//小程序从前台进入后台时触发。

console.log('app >> onHide');

},

onError:function(error){

// 小程序发生脚本错误，或者api调用失败时触发。

// error String 错误信息，包含堆栈信息

console.log('app >> onError ， error::'+error);

},

onPageNotFound(Object){

// 基础库1.9.90开始支持

// 小程序要打开的页面不存在时触发。

// Object参数说明：

// path String 不存在的页面路径

// query object 打开不存在页面的query

// isEntryPage Boolean 是否本次启动的首个页面

console.log('app >> onPageNotFound , Object :: ',Object);

//注：如果开发者没有添加 onPageNotFound 监听，当跳转页面不存在时，将推入微信客户端原生的页面不存在提示页面。

// 如果 onPageNotFound 回调中又重定向到另一个不存在的页面，将推入微信客户端原生的页面不存在提示页面，并且不再回调 onPageNotFound。

}

})

## 触摸事件（滑动等）

首先在需要滑动的元素上添加bindtouchstart、bindtouchend和bindtouchmove事件

Bindtouchstart在触摸屏幕时启动，记录触摸点坐标

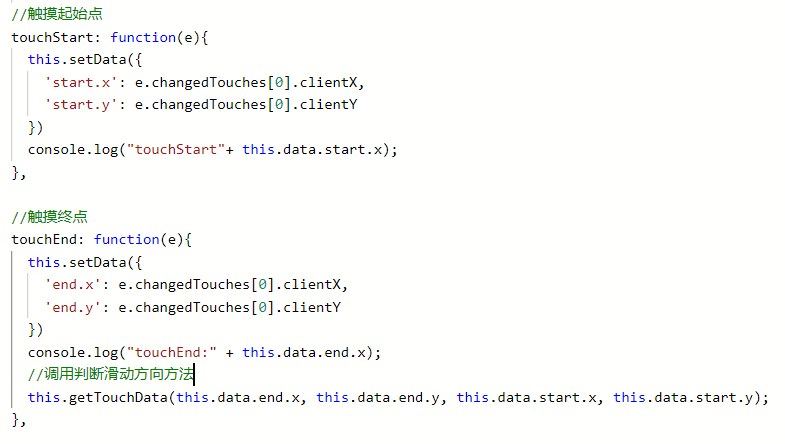
Bindtouchend在离开屏幕时启动，记录终结点坐标

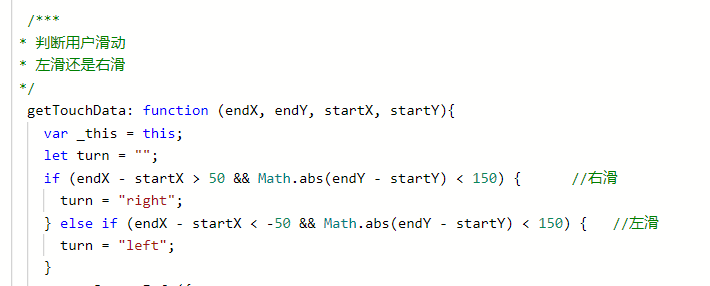
Bindtouchmove在触摸屏幕时启动，实时记录触摸点移动情况（距离）

Wxml:

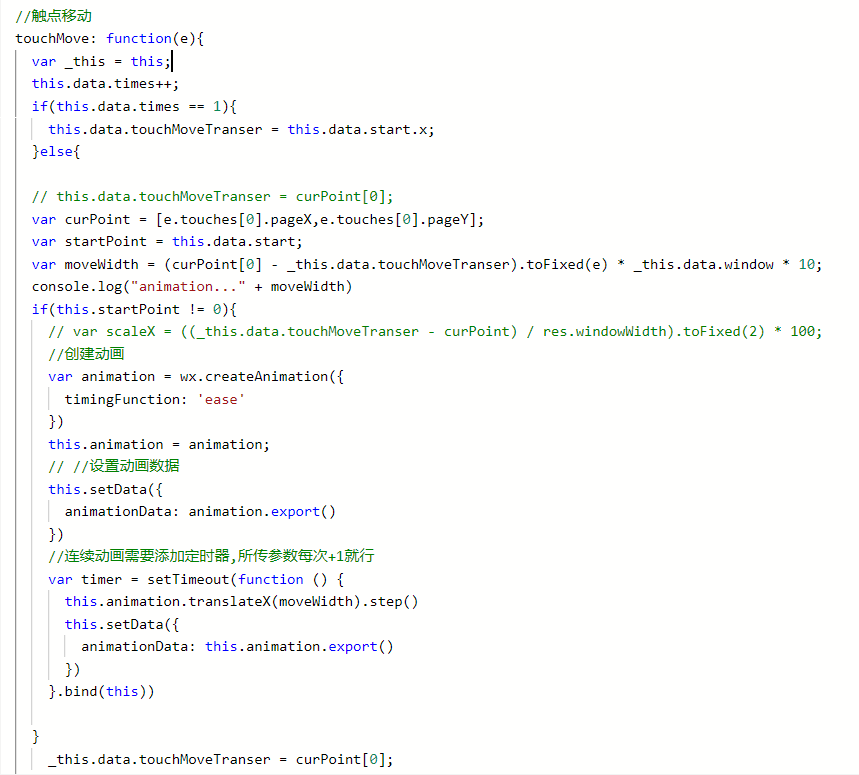


Js:



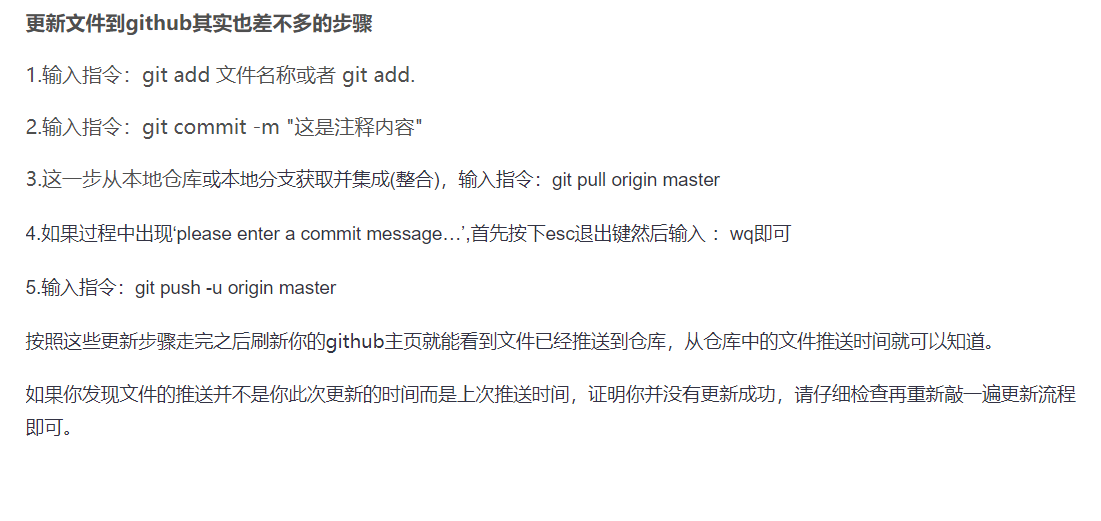


手指滑动移动图片（但是触摸滑动不太精确，建议使用swiper插件）：



在线预览github项目页面方法：<https://htmlpreview.github.io/?+（github>页面链接）

本地内容更新到github方法



## 兼容性

1. 手机端opacity无效
2. pageLifetimes周期仅安卓手机可使用