目录

**[一、基本文件详解](#_Toc6759_WPSOffice_Level1)** **[1](#_Toc6759_WPSOffice_Level1)**

[1、app.js](#_Toc25682_WPSOffice_Level2) [1](#_Toc25682_WPSOffice_Level2)

[2、app.wxss](#_Toc4654_WPSOffice_Level2) [1](#_Toc4654_WPSOffice_Level2)

[3、 app.json](#_Toc27085_WPSOffice_Level2) [1](#_Toc27085_WPSOffice_Level2)

[4、page相关](#_Toc11914_WPSOffice_Level2) [2](#_Toc11914_WPSOffice_Level2)

**[二、基本操作详解](#_Toc25682_WPSOffice_Level1)** **[2](#_Toc25682_WPSOffice_Level1)**

[各基本标签使用注意](#_Toc28662_WPSOffice_Level2) [2](#_Toc28662_WPSOffice_Level2)

[onclick事件绑定及传参](#_Toc25979_WPSOffice_Level2) [3](#_Toc25979_WPSOffice_Level2)

[获取input的值](#_Toc23938_WPSOffice_Level2) [3](#_Toc23938_WPSOffice_Level2)

[wxml的引入](#_Toc6618_WPSOffice_Level2) [3](#_Toc6618_WPSOffice_Level2)

[全局变量](#_Toc11511_WPSOffice_Level2) [4](#_Toc11511_WPSOffice_Level2)

[For循环](#_Toc15074_WPSOffice_Level2) [5](#_Toc15074_WPSOffice_Level2)

[ajax数据请求](#_Toc7240_WPSOffice_Level2) [5](#_Toc7240_WPSOffice_Level2)

[自定义组件component](#_Toc29654_WPSOffice_Level2) [6](#_Toc29654_WPSOffice_Level2)

**[三、综合实例](#_Toc4654_WPSOffice_Level1)** **[7](#_Toc4654_WPSOffice_Level1)**

[1、轮播](#_Toc25182_WPSOffice_Level2) [7](#_Toc25182_WPSOffice_Level2)

[2、for循环出列表，点击一条数据跳页传参](#_Toc15840_WPSOffice_Level2) [8](#_Toc15840_WPSOffice_Level2)

[3、下拉刷新](#_Toc25084_WPSOffice_Level2) [8](#_Toc25084_WPSOffice_Level2)

[4、 侧滑菜单](#_Toc3572_WPSOffice_Level2) [11](#_Toc3572_WPSOffice_Level2)

**[四、官方接口](#_Toc27085_WPSOffice_Level1)** **[13](#_Toc27085_WPSOffice_Level1)**

[1、系统信息](#_Toc19607_WPSOffice_Level2) [13](#_Toc19607_WPSOffice_Level2)

[2、 热更新](#_Toc26297_WPSOffice_Level2) [14](#_Toc26297_WPSOffice_Level2)

[3、 路由](#_Toc32029_WPSOffice_Level2) [14](#_Toc32029_WPSOffice_Level2)

[4、 调取相册相机](#_Toc5830_WPSOffice_Level2) [15](#_Toc5830_WPSOffice_Level2)

[5、扫描二维码](#_Toc20925_WPSOffice_Level2) [15](#_Toc20925_WPSOffice_Level2)

[5、指纹解锁](#_Toc28082_WPSOffice_Level2) [16](#_Toc28082_WPSOffice_Level2)

**[五、 自定义组件](#_Toc11914_WPSOffice_Level1)** **[16](#_Toc11914_WPSOffice_Level1)**

[1、wxchart](#_Toc17713_WPSOffice_Level2) [17](#_Toc17713_WPSOffice_Level2)

[2、qrcode二维码](#_Toc18127_WPSOffice_Level2) [17](#_Toc18127_WPSOffice_Level2)

**[六、 微信自带云服务](#_Toc28662_WPSOffice_Level1)** **[18](#_Toc28662_WPSOffice_Level1)**

[初始化](#_Toc19225_WPSOffice_Level2) [18](#_Toc19225_WPSOffice_Level2)

[新增](#_Toc32385_WPSOffice_Level2) [18](#_Toc32385_WPSOffice_Level2)

[查询](#_Toc20580_WPSOffice_Level2) [19](#_Toc20580_WPSOffice_Level2)

[更新与删除](#_Toc20373_WPSOffice_Level2) [21](#_Toc20373_WPSOffice_Level2)

# 一、基本文件详解

1、app.js

全局变量 全局方法

onLaunch 初始化完成后只会触发一次

onShow 后台进入前台触发

onHide 前台进入后台

globalData 全局变量

2、app.wxss

全局CSS样式文件

1. app.json

{

"pages": ["pages/index/index", "pages/logs/index"],

"window": {

"navigationBarTitleText": "Demo"

},

"tabBar": {

"list": [

{

"pagePath": "pages/index/index",

"text": "首页"

},

{

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

"text": "日志"

}

]

},

"networkTimeout": {

"request": 10000,

"downloadFile": 10000

},

"debug": true,

"navigateToMiniProgramAppIdList": ["wxe5f52902cf4de896"]

}

"navigationBarBackgroundColor": "#ff4f02", //标题栏背景色

"navigationBarTitleText": "来云游", //标题

networkTimeout 超时

debug debug模式

tabBar 底部选项卡

## 4、page相关

data [Object] 页面的初始数据

onLoad [function] 页面加载 一个页面只会调用一次

onShow [function] 页面显示 每次打开页面都会调用一次

onReady [function] 页面初次渲染完成 一个页面只会调用一次 对界面的设置如wx.setNavigationBarTitle请在onReady之后设置

onHide [function] 页面隐藏 当navigateTo或底部tab切换时调用

onUnload [function] 页面加载 当redirectTo或navigateBack的时候调用

onPullDownRefresh: 下拉刷新

监听用户下拉刷新事件。

需要在config的window选项中开启enablePullDownRefresh。

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: 上拉触底

监听用户下拉触底事件。

onPageScroll: 页面滚动

监听用户滑动页面事件。

参数为 Object，包含以下字段：

onShareAppMessage: 用户转发

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

用户点击转发按钮的时候会调用

此事件需要 return 一个 Object，用于自定义转发内容

# 二、基本操作详解

TypeScript转JavaScript

//todo

## 各基本标签使用注意

数据绑定 {{变量名}}

view 相当于div

text 只有在text里边的文字可以选中

scroll-view 需要先给 scroll-y=‘true’ 不然也不会滚动

rich-text的使用

<rich-text nodes="{{nodes}}" bindtap="tap"></rich-text>

p标签等不能写成< p >

## onclick事件绑定及传参

bindtap=’方法名’

在wxml页面上传参的时候，加data-[参数名]，有多个的话写多次

方法里加变量e

function(e){ e.currentTarget.dataset.[变量名] //获取方式 }

## 获取input的值

<input bindinput ="方法名"/>

js中

data: {

变量:"",

}

方法名:function (e) {

this.setData({

变量: e.detail.value

})

},

其它方法中通过this.data.pwdA 调取

## wxml的引入

### import

单wxml

<template name="item">

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

</template>

一个wxml页面可以写多个template

template 中的name是关键 后边通过is 调取

<!--引入到的page的wxml -->

<import src="item.wxml" />

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

is为模板名 data 里边传参数

### include

<!-- index.wxml -->

<include src="header.wxml" />

<!-- header.wxml -->

<view>header</view>

相当于是把整个代码 拷贝到 include 位置

全局变量

### globalData 方法

app.js 里的globalData下写变量/方法

使用页

const app = getApp()

app.globalData.[变量/方法名]

### util方法

utils/util.js （没有自己建）

定义变量/方法

注册指令

module.exports = {

showpic: showpic,

HTTP:HTTP，

变量名：变量/方法

}

使用页

const util = require('../../utils/util.js'); //引入

util.方法名(参数)

util.变量

备注：貌似变量及方法无法在wxml中直接使用，只能再中转一次

### 更新及调取

更新较为简单：

util.HTTP="新值"

app.globalData.ImgUrl = "新值";

本页面及其它页面需要更新值

this.setData({

HTTP: util.HTTP

})

this.setData({

ImgUrl: app.globalData.ImgUrl

})

## For循环

<view wx:for="{{list}}" data-item="item" wx:for-index="idx">

第{{idx+1}}次输出{{item.aa}}

</view>

wx:for为循环的对象

data-item为循环对象在循环中的名称

wx:for-index为index

## ajax数据请求

取消域名效验

右上脚 详情

选中 - 不校验合法域名、web-view（业务域名）、TLS 版本以及 HTTPS 证书

myajax: function (){

var that = this;

wx.request({

url: 'https://easy-mock.com/mock/5cbe7776b28a7920ecb1f5e4/example/mock',

//仅为示例，并非真实的接口地址

data: {

Data1 : '',

Data2 : ''

},

header: {

'content-type': 'application/json' // 默认值

},

success: function (res) {

console.log(res.data.data.projects)

that.setData({

list: res.data.data.projects

});

//list为之前定义的变量，setData此处是为了重赋值以刷新页面

}

})

},

## 自定义组件component

组件的创建 右键 新建Component

定义可以传到自定义组件中的变量

properties: {

content: {

type: String,

value: '变量'

},

},

使用页

Json中引入组件

{

"usingComponents": {

"example": "/pages/component/example"

}

}

Wxml中引入内容

<example id="example"> </example>

id是后边用来调用组件中方法的,example 为组件名

Js

onReady: function () {

//获得example组件

this.example = this.selectComponent("#example");

},

在方法中执行

This.组件名.方法名()

### solt插槽的使用

Component下

options: {

multipleSlots: true // 在组件定义时的选项中启用多slot支持

},

//todo

# 三、综合实例

## 1、轮播

wxml页面

<swiper indicator-dots="{{indicatorDots}}"

autoplay="{{autoplay}}" interval="{{interval}}" duration="{{duration}}" circular="true">

<block wx:for="{{arr}}">

<swiper-item>

<image src="{{item}}" class="slide-image" width="355" height="150"/>

</swiper-item>

</block>

</swiper>

备注 ：这段代码加在view之外 后边的slide-image 为class ，重新定义下

index.js页面

data: {

mode: "scaleToFill",

arr: [],

indicatorDots: true,

autoplay: true,

interval: 3000,

duration: 1000,

},

var array = this.data.arr

for (let i = 1; i < 3; i++) {

array.push("../../assets/images/swiper" + i + ".png")

}

this.setData({ arr: array })

加入对应图片

## 2、for循环出列表，点击一条数据跳页传参

List页

Wxml

<view wx:for="{{list}}" data-item="item" wx:for-index="idx" bindtap='linkShows' wx:key="\*this" data-id="{{item.url}}">

<text >{{idx+1}}{{item.address}}推荐指数{{item.string+item.url}}</text>

</view>

Js

linkShows: function (event){

var id = event.currentTarget.dataset.id

console.log(id)

wx.navigateTo({

url: '/pages/show/show?id=' + id +'&key=123'　　// 页面 A

})

}

show页

this.setData({

msg: options.id

})

msg为之前定义的变量 options.[变量名]即可获得传过来的参数

## 3、下拉刷新

wxml

<scroll-view class="dropList" scroll-y="true" bindscrolltoupper="refresh" bindscrolltolower="loadMore">

<view class="dropList\_tip" hidden="{{hideHeader}}">

<view>下拉刷新中……</view>

<view>{{refreshTime}}</view>

</view>

<view class="dropListItem" wx:for="{{contentlist}}" wx:key="index" wx:for-index="idx" >

<view class="dropListItem\_tit">{{idx+1}}、{{item.ttnm}}</view>

</view>

<view class="dropList\_tip" hidden="{{hideBottom}}">

<view>{{loadMoreData}}</view>

<view>{{refreshTime}}</view>

</view>

</scroll-view>

Js

data: {

hideHeader: true,

hideBottom: true,

refreshTime: '', // 刷新的时间

contentlist: [], // 列表显示的数据源

allPages: '', // 总页数

currentPage: 1, // 当前页数 默认是1

loadMoreData: '加载更多……' ,

},

onLoad: function (options) {

var that=this;

var date = new Date();

this.setData({

refreshTime: date.toLocaleTimeString()

})

console.log("当前刷新时间" + that.refreshTime)

this.getData();

},

// 上拉加载更多

loadMore: function () {

console.log('上拉加载更多');

var self = this;

// 当前页是最后一页

if (self.data.currentPage == self.data.allPages) {

self.setData({

loadMoreData: '已经到顶'

})

return;

}

setTimeout(function () {

console.log('上拉加载更多');

var tempCurrentPage = self.data.currentPage;

tempCurrentPage = tempCurrentPage + 1;

self.setData({

currentPage: tempCurrentPage,

hideBottom: false

})

self.getData();

}, 1000);

},

// 下拉刷新

refresh: function (e) {

var self = this;

setTimeout(function () {

console.log('下拉刷新');

var date = new Date();

self.setData({

currentPage: 1,

refreshTime: date.toLocaleTimeString(),

hideHeader: false

})

self.getData();

},1000);

},

// 获取数据 pageIndex：页码参数

getData: function () {

var that = this;

var pageIndex = that.data.currentPage;

wx.request({

url: 'http://hzz.test.lonhcloud.cn/lhHzzTtcServerlet/commonServerlet',

data: {

prj:'lhHzzTtcManager',

bean: 'TtcManagerSS',

method: 'findTasktargetbPageListByOrder',1: pageIndex,2: '12',3: '-1',4: '13871',5: '1,2,3',6: '',7: '-1',8: '',9: '',},

success: function (res) {

console.log("进了")

var dataModel = res.data;

for (let i = 0; i < dataModel.dataList.length; i++) {

console.log(dataModel.dataList[i]["systm"])

dataModel.dataList[i]["systm"] = dataModel.dataList[i]["systm"].substring(0,11)

console.log(dataModel.dataList[i]["systm"])

}

console.log(dataModel.dataList)

if (pageIndex == 1) { // 下拉刷新

that.setData({

allPages: dataModel.recordNumber,

contentlist: dataModel.dataList,

hideHeader: true

})

} else { // 加载更多

console.log('加载更多');

var tempArray = that.data.contentlist;

tempArray = tempArray.concat(dataModel.dataList);

that.setData({

allPages: dataModel.recordNumber,

contentlist: tempArray,

hideBottom: true

})

}

},

fail: function () {

}

})

},

## 侧滑菜单

wxml

<view class="page" bindtouchend='touchend' bindtouchstart='touchstart'>

<!-- 侧滑栏 -->

<view class="page\_tab">

<view >

<!--这里边放菜单 -->

<navigator url='../map/map'>

<view class="wc">

<text>订单记录</text>

</view>

</navigator>

</view>

</view>

<!-- 侧滑栏背景 -->

<view class="bg" bindtap='hideview' style='display:{{display}}'></view>

<!-- 主页 -->

<view class="home" style="{{translate}}">

<!--这里边放正文的内容 -->

<text bindtap='showview' > 点我出菜单</text>

</view>

</view>

Css

page{

height: 100%

}

/\* 主页 \*/

.home {

position: absolute;

width: 750rpx;

height: 100%;

background-color: white;

z-index: 1;

transition: All 0.4s ease;

-webkit-transition: All 0.4s ease;

}

/\* 遮罩层 \*/

.bg {

display: none;

position: fixed;

top: 0%;

left: 70%;

width: 100%;

height: 100%;

background-color: black;

z-index: 1001;

-moz-opacity: 0.7;

opacity: 0.70;

transition:width 2s;

filter: alpha(opacity=70);

}

/\* 侧滑栏 \*/

.page\_tab {

height: 100%;

width: 750rpx;

position: fixed;

background-color: white;

z-index: 0;

}

Js

var start\_clientX;

var end\_clientX;

Page({

data: {

windowWidth: wx.getSystemInfoSync().windowWidth

},

// 滑动开始

touchstart: function (e) {

start\_clientX = e.changedTouches[0].clientX

},

// 滑动结束

touchend: function (e) {

end\_clientX = e.changedTouches[0].clientX;

if (end\_clientX - start\_clientX > 120) {

this.setData({

display: "block",

translate: 'transform: translateX(' + this.data.windowWidth \* 0.7 + 'px);'

})

} else if (start\_clientX - end\_clientX > 0) {

this.setData({

display: "none",

translate: ''

})

}

},

// 弹出菜单

showview: function () {

this.setData({

display: "block",

translate: 'transform: translateX(' + this.data.windowWidth \* 0.7 + 'px);'

})

},

// 隐藏菜单

hideview: function () {

this.setData({

display: "none",

translate: '',

})

}

})

## 腾讯地图(只提供导航功能)

app.json

"plugins": {

"myPlugin": {

"version": "1.0.6",

"provider": "wx5bc2ac602a747594"

}

Wxml

<map-route class="mymap" route-info="{{routeInfo}}"></map-route>

Js

routeInfo: {

startLat: 0.0, //起点经度 选填

startLng: 0.0, //起点纬度 选填

startName: "未获取到起点", // 起点名称 选填

endLat: 25.0585733749, // 终点经度必传

endLng: 102.6621401310, //终点纬度 必传

endName: "和成国际", //终点名称 必传

mode: "car" //算路方式 选填

}

# 四、官方接口

## 1、系统信息

wx.getSystemInfo({

success(res) {

console.log("getSystemInfo")

console.log(res)

}

})

try {

const res = wx.getSystemInfoSync()

console.log("getSystemInfoSync")

console.log(res)

} catch (e) {

// Do something when catch error

}

getSystemInfo与getSystemInfoSync获取的信息一致

## 热更新

getLaunchOptionsSync

//todo

## 路由

**wx.switchTab** 跳转到 tabBar 页面，并关闭其他所有非 tabBar 页面

**wx.reLaunch** 关闭所有页面，打开到应用内的某个页面

**wx.redirectTo** 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。

**wx.navigateTo** 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/wx.navigateBack.html) 可以返回到原页面。小程序中页面栈最多十层。

（列表到详情页使用这种）

示例：这4个都可以用这种结构

wx.navigateTo({

url: '/pages/main/main'　　// 页面 A

})

**[wx.navigateBack](https://developers.weixin.qq.com/miniprogram/dev/api/wx.navigateBack.html)**

关闭当前页面，返回上一页面或多级页面。可通过 [getCurrentPages](https://developers.weixin.qq.com/miniprogram/dev/reference/api/getCurrentPages.html) 获取当前的页面栈，决定需要返回几层。

示例：

wx.navigateBack({

delta: 2

})

## 调取相册相机

wx.chooseImage({

count: 1,

sizeType: ['original', 'compressed'],

sourceType: ['album', 'camera'],

success(res) {

// tempFilePath可以作为img标签的src属性显示图片

const tempFilePaths = res.tempFilePaths

console.log(tempFilePaths )

}

})

}

## 5、扫描二维码

data中定义sweepCodeShow

var that = this;

var show;

wx.scanCode({

success: (res) => {

this.show = res.result;

this.setData({

sweepCodeShow: this.show

})

wx.showToast({

title: '成功',

icon: 'success',

duration: 2000

})

},

fail: (res) => {

wx.showToast({

title: '失败',

icon: 'success',

duration: 2000

})

},

complete: (res) => {

}

})

因为涉及到setData 所以这个方法无法公用，不知道未来能不能解决

## 5、指纹解锁

if (wx.canIUse('checkIsSupportSoterAuthentication')) { // 检测当前微信版本是否支持调用指纹相关接口

wx.checkIsSupportSoterAuthentication({

success(res) {

if (res.supportMode[0] == 'fingerPrint') { // 写法不严谨，正确写法应该是遍历数组查找

// 在此具体调用指纹识别

console.log("指纹识别")

console.log(res)

wx.startSoterAuthentication({

requestAuthModes: ['fingerPrint'],

challenge: '123456',

authContent: '请用指纹解锁',

success(res) {

// res 中包含授权数据，需要进一步验证正确性

},

fail(res) {

console.log('用户取消了指纹识别，或调用出现错误');

wx.redirectTo({

url: '/pages/index/index',

})

}

})

} else {

console.log('当前设备不支持指纹识别')

}

}

})

}

# 自定义组件

## 1、wxchart

GIT: <https://github.com/xiaolin3303/wx-charts>

dist/wxcharts.js 复制到项目的utils文件夹下

wxml页面：<canvas canvas-id="canvas1"></canvas>

js页面：

引入：const wxcharts = require('../../utils/wxcharts.js')

onload方法中调用：

new wxcharts({

canvasId: 'canvas1',

type: 'pie',

series: [{ name: '一班', data: 50 }, { name: '二班', data: 30 }, { name: '三班', data: 20 }, { name: '四班', data: 18 }, { name: '五班', data: 8 }],

width: 412,

height: 412,

dataLabel: true,

});

支持图表类型 饼图pie 圆环图ring 线图line 柱状图column 区域图area 代码分析 Here

## 2、[qrcode](https://github.com/demi520/wxapp-qrcode)二维码

GIT: <https://github.com/tomfriwel/weapp-qrcode>

dist/weapp-qrcode.js 复制到项目的utils文件夹下

wxml页面：

<canvas class='canvas' canvas-id='canvas'></canvas>

<button bindtap='tapHandler' data-code='123abc456efg000'>转为二维码</button>

js页面

引入：

var QRCode = require('../../../utils/weapp-qrcode.js');

var qrcode;

onload方法中调用：

var res = wx.getSystemInfoSync()

console.log(res.windowWidth)

qrcode = new QRCode('canvas', {

text: "code=0000000000000",

width: res.windowWidth,

height: res.windowWidth,

colorDark: "#000000",

colorLight: "#ffffff",

correctLevel: QRCode.CorrectLevel.H,

});

tapHandler: function (e) {

qrcode.makeCode(e.target.dataset.code); //用元素对应的code更新二维码

},

其它二维码工具

GIT: <https://github.com/demi520/wxapp-qrcode>

utils/qrcode.js 复制到项目的utils文件夹下

# 微信自带云服务

## 初始化

wx.cloud.init({

traceUser: true,

env: 'yargs-jsdemo01' //云服务ID

})

const db = wx.cloud.database();

const \_ = db.command //这句是后边的查询指令

## 新增

db.collection('user').add({

// data 字段表示需新增的 JSON 数据

data: {

// \_id: 'todo-identifiant-aleatoire', // 可选自定义 \_id，在此处场景下用数据库自动分配的就可以了

Num:1,

title:"标题2",

anthor:"作者是谁",

description: '插入数据库2',

systm: new Date(),

tags: [

'cloud',

'database'

],

// 为待办事项添加一个地理位置（113°E，23°N）

location: new db.Geo.Point(113, 23),

done: false

},

success(res) {

// res 是一个对象，其中有 \_id 字段标记刚创建的记录的 id

console.log(res)

}

})

.where({

\_openid: 'osu\_r4vpx82ry4uPP94YyaPBQvbM'

})

## 查询

查全表

db.collection('user').get(

{

success(res) {

console.log(res.data)

}

}

)

按条件查

db.collection('user').where({

age: \_.gt(18)

}).get(

{

success(res) {

// res.data 是包含以上定义的两条记录的数组

console.log("标题")

console.log(res.data)

}

}

)

eq 等于

neq 不等于

lt 小于

lte 小于或等于

gt 大于

gte 大于或等于

in 字段值在给定数组中

nin 字段值不在给定数组中

有 and，也有 or

### 升降序

在表名后，where前

.orderBy('Num', 'desc')

### 限制调取多少条

.limit(5)

跳过几条（分页时用）

.skip(2)

### 共有多少条数据

.count({

success: function (res) {

console.log("一共有多少条数据"+res.total);

}

})

## 更新与删除

doc里的指为ID

db.collection('user').doc('96c1cbbe5cc2aed506eb4bfa07b7928d')

.update({

data: {

done: true

}

})

多条更新有文档但无法实现

删除

.remove()

多条删除有文档但无法实现