

小程序技术文档

一 . 小程序应用分析

1. 简单介绍

小程序是一种微信开发不需要下载安装即可使用的应用。

2. 实质

微信小程序就是 Hybrid 技术的应用。

Hybrid App(混合模式移动应用)。

小程序能够更多的可以更多的调用手机本身的功能（如位置信息，摄像头等）。

3. 小程序注册

小程序开发框架的逻辑层是由 JavaScript 编写。逻辑层将数据进行处理后发送给视图层，同时接受视图层的事件反馈。小程序再次基础上做了一些修改方便开发：增加 App 和 Page 方法，进行程序和页面的注册。

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。

属性	类型	描述	触发时机
onLaunch	Function	生命周期函数--监听小程序初始化	当小程序初始化完成时，会触发 onLaunch（全局只触发一次）
onShow	Function	生命周期函数--监听小程序显示	当小程序启动，或从后台进入前台显示，会触发 onShow
onHide	Function	生命周期函数--监听小程序隐藏	当小程序从前台进入后台，会触发 onHide
其他	Any	开发者可以添加任意的函数或数据到 Object 参数中，用 this 可以访问	

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。生命周期函数为：

onLoad: 页面加载

一个页面只会调用一次。

onShow: 页面显示

每次打开页面都会调用一次。

onReady: 页面初次渲染完成

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

onHide: 页面隐藏

当 navigateTo 或底部 tab 切换时调用

onUnload: 页面卸载

当 redirectTo 或 navigateBack 的时候调用

二. 小程序实例解析

1. 创建项目

现在是内测版本，所有的 AppID 全部都是腾讯内部发布的。但是，没有 AppID 也不影响测试

开发，我们可以选择无 AppID 进行测试开发，只是不能在手机真机上面调试。



< 返回

添加项目

AppID 无 AppID 部分功能受限

返回填写小程序AppID

项目名称 小程序

项目目录 选择

取消 添加项目

选择项目目录，再添加项目即可。

2. 编写代码

点击开发者工具左侧导航的「编辑」，我们可以看到这个项目，已经初始化并包含了一些简单的代码文件。是 **app.js**、**app.json**、**app.wxss** 这三个。其中 **app.js** 是我们传统的 js 文件，**app.json** 是项目配置文件，**app.wxss** 是项目 css 文件，微信小程序会读取这些文件，并生成小程序实例。

(1)app.json:

app.json 是对整个小程序的全局配置。其中有 5 个属性，官方给出的配置表为：

属性	类型	必填	描述
<code>pages</code>	String Array	是	设置页面路径
<code>window</code>	Object	否	设置默认页面的窗口表现
<code>tabBar</code>	Object	否	设置底部 tab 的表现
<code>networkTimeout</code>	Object	否	设置网络超时时间
<code>debug</code>	Boolean	否	设置是否开启 debug 模式

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。

window 是用于设置小程序的状态栏、导航条、标题、窗口背景色。

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	下拉背景字体、loading 图的样式，仅支持 dark/light
enablePullDownRefresh	Boolean	false	是否开启下拉刷新，详见 页面相关事件处理函数 。

pages 里面是程序的所有页面的目录，所有需要跳转的页面，都需要在 **pages** 里面配置好。

(2)tabBar:

tabBar 是底部导航栏部分，**tabBar** API 为

属性	类型	必填	默认值	描述
color	HexColor	是		tab 上的文字默认颜色
selectedColor	HexColor	是		tab 上的文字选中时的颜色
backgroundColor	HexColor	是		tab 的背景色
borderStyle	String	否	black	tabbar上边框的颜色， 仅支持 black/white
list	Array	是		tab 的列表，详见 list 属性说明，最少2个、最多5个 tab
position	String	否	bottom	可选值 bottom、top

tabBar 配置好后，在任何页面下，都会有一个 tab 导航栏，其中 list 里面是配置 tab 里有多少个按钮,案例中为两个。list 里面有多个属性，

属性	类型	必填	说明
pagePath	String	是	页面路径，必须在 pages 中先定义
text	String	是	tab 上按钮文字
iconPath	String	是	图片路径，icon 大小限制为40kb
selectedIconPath	String	是	选中时的图片路径，icon 大小限制为40kb

案例 APP 的 app.json 为：

```
{
  "pages": [
    "pages/index/index",
    "pages/logs/logs"
  ],
  "window": {
    "backgroundTextStyle": "light",
    "navigationBarBackgroundColor": "green",
    "navigationBarTitleText": "APP",
    "navigationBarTextStyle": "white"
  },
  "tabBar": {
    "selectedColor": "red",
    "list": [
      {
        "pagePath": "pages/index/index",
        "text": "首页",
```

```

        "iconPath": "goods_mgold.png",

        "selectedIconPath": "goods_mgold.png"
    }, {

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

        "text": "天气查询",

        "iconPath": "icon_community.png",

        "selectedIconPath": "icon_community.png"
    }
  ]
}

```

上面那段配置代码展示出来的效果是：



(3)wxml 文件：

微信的 wxml 文件相当于传统的 html 文件，省去了一些微信 APP 开发不需要的标签，如 H1-H5，用了这些就会报错，其中 html 中的 div 标签，在微信中变成了 view 标签。（也就是换了个名字。。。）

(4)app.js:

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 MINA 提供的丰富的 API。代码主要是写在 APP 对象里面作用于全局。

其中每个页面都可以有自己的 js 文件，例如 index.js 就是 Index.wxml 页面的 js 代码，其中 js 代码的一些应用主要是写在 page 对象里面。

事件的使用方式:

首先在 wxml 里面写入一个 bindtap 点击事件。

```
<view id="tapTest" data-hi="WeChat" bindtap="tapName"> Click me! </view>
```

然后再 js 的 page 对象中定义：

```
Page({
  tapName: function(event) {
    console.log(event)
  }
})
```

就可以实现一个点击事件。其中 bind 是绑定，type 为 tap。type 是事件类型。

数据渲染:

在组件上使用 wx:for 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

xwml 里面写入：

```
<view wx:for="{{array}}">
  {{index}}: {{item.message}}
</view>
```

index.js 里面写入：

```
Page({
  data: {
    array: [{
      message: 'foo',
    }, {
      message: 'bar'
    }]
  }
})
```

条件渲染:

wx:if 来判断是否在页面是进行渲染显示

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

可以在 Page 对象里面的 data 属性里面写入 condition 的值为 true 或者 false 判断是否渲染。

模板定义:

可以在模板中定义代码片段，然后在不同的地方调用。

直接在外部新建一个 box.wxml 的模板：


```

<scroll-view>
  <text>{{site}}</text>
  <view wx:if="{{weather=='晴'}}" class="icoImg sun"></view>
  <view wx:elif="{{weather=='多云'}}" class="icoImg m"></view>
  <view wx:elif="{{weather=='阴转多云'}}" class="icoImg m"></view>
  <view wx:elif="{{weather=='多云转阴'}}" class="icoImg m"></view>
  <view wx:elif="{{weather=='阴转多云'}}" class="icoImg m"></view>
  <view wx:elif="{{weather=='多云转晴'}}" class="icoImg m-sun"></view>
  <view wx:elif="{{weather=='多云转阵雨' || weather=='阵雨转多云'}}" class="icoImg m-rain"></view>
  <view wx:elif="{{weather=='中雨转多云'}}" class="icoImg m-rain"></view>
  <view wx:elif="{{weather=='小雨'}}" class="icoImg rain"></view>
  <text>{{weather}}</text>
  <text>{{wind}}</text>
  <text>{{week}}</text>
  <text>{{advice}}</text>
</scroll-view>

```

然后建立一个外部 commom.js 模块。

```

function show(city,that){
wx.request({
  url: 'http://v.juhe.cn/weather/index?callback=?',
  data: {
    cityname: city ,
    key: '973839fb09bc6cadf575f9877fab98ef'
  },
  header: {
    'Content-Type': 'application/json'
  },
  success: function(res) {

    var arr =[];
    var today_data = res.data["result"]["today"];
    var future = res.data["result"]["future"];
    console.log(res)
    for(var i in future){
      arr.push(future[i]);
    }
    console.log(arr[0]);
    that.setData({
      f_data : arr,
      site : today_data.city,
      day:today_data.date_y,
      advice:today_data.dressing_advice,
      weather:today_data.weather,
      week:today_data.week,
      wind:today_data.wind,
    })

  },

  })
}
module.exports = {
  show:show
}

```

通过 module.exports 导出模块，

先在需要引入模块的 wxml 文件中直接 include 带上 src 地址

```
<include src="../../box.wxml"/>
```

然后在需要引入模块的 js 文件中:

```
var common = require('../../common.js')
```

然后用 common.show()调用。

这样就可以复用这个模块了。在任何页面中只需要用 include 导入 wxml 代码，用 require 引入 js 文件就可以添加这个模块。



(5)wxss:

wxss 文件就是传统的 css 文件，没有很大的区别。

但是其中微信给一套响应式的布局

rpx (responsive pixel)：可以根据屏幕宽度进行自适应。规定屏幕宽为 750rpx。如在 iPhone6 上，屏幕宽度为 375px，共有 750 个物理像素，则 $750\text{rpx} = 375\text{px} = 750$ 物理像素， $1\text{rpx} = 0.5\text{px} = 1$ 物理像素。

rpx 的原理就是 rem 布局原理。只是换个名字，少了一步屏幕 fon-size 换算的 Js 代码，微信在内部执行了，不需要自己写了。

(6)接口 API:

小程序开发框架提供丰富的微信原生 API，可以方便的调起微信提供的的能力，如获取用户信息，本地存储，支付功能等。

API 文档地址

[https://mp.weixin.qq.com/debug/wxadoc/dev/api/api-network.html?
t=1477656494973](https://mp.weixin.qq.com/debug/wxadoc/dev/api/api-network.html?t=1477656494973)