**微信小程序项目框架介绍**

中国人民大学

信息学院

李信阔(2017104120)

2018.10.14

**微信小程序中只有四种类型的文件**

* js-----------JavaScript文件
* json---------项目配置文件，负责窗口的颜色等等
* wxml---------类似html文件
* wxss---------类似CSS文件

在根目录下用app来命名的这四种类型的文件，就是程序入口文件。

**app.json**

必须要有这个文件，如果没有这个文件，IDE就会报错，因为微信框架把这个作为配置文件入口，我们只需要创建这个文件，在里面写个大括号就行。以后我们会在这里对整个小程序的全局配置。记录了页面的组成，配置小程序的窗口、背景、配置导航条的样式，配置默认标题。

**app.js**

必须要有这个文件，没有也是会报错的，但是这个文件创建一下就行，什么都不用写，以后我们可以在这个文件中监听并且处理小程序的生命周期函数、声明全局变量。

**app.wxss**

这个文件不是必须的。一位内他只是全局的css文件的样式。

**app.wxml**

这个也不是必须的，而且这个指的并不是主界面，因为小程序中的主界面是靠JSON文件中的配置类决定的。

**有了app.js,app.json这两个文件在运行程序的时候IDE就不会报错了，也意味着这就是最简单的微信小程序**

**Hello Word**

**创建程序实例**

app.js文件管理整个程序的生命周期，所以在里面添加如下的代码：（输入App IDE会有提示）

App({

onlaunch: function() {

console.log('App Launch')

},

onShow: function() {

console.log('App Show')

},

onHide: function() {

console.log('App Hide')

}

})

**具体的API解释如下**

**App()**

App() 函数用来注册一个小程序。接受一个Object参数，其制定小程序的生命周期函数。

**object参数说明**

| **属性** | **类型** | **描述** | **触发时机** |
| --- | --- | --- | --- |
| onLaunch | Function | 生命周期函数--监听小程序的初始化 | 当小程序初始化完成时，会出发**onLaunch**（全局只触发一次） |
| onShow | Function | 生命周期函数--监听小程序的显示 | 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--监听小程序的隐藏 | 当小程序从前台进入后台会出发**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 |  |

**前台、后台定义：** 当用户点击左上角关闭，或者按了设备的Home键离开温馨，小程序并没有直接销毁，而是进入了后台；当再次打开微信或者进入小程序的时候，又会从后台进入前台。只有当小程序进入后台一定的时间，或者系统资源占用过高，才会被真正的销毁。

**美化ActionBar**

json文件负责配置ActionBar颜色，我们只需要在里面添加如下代码即可

**app.json配置项列表**

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

| **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 |

**美化界面**

美化界面用到的是wxml、wxss文件，为了让程序代码结构简洁我们需要在根目录下创建一个新文件夹，名字随意，这里我们叫pages，然后我们在pages文件夹里面在创文件夹，名字随意，这里我们叫index  
然后我们创建index.wxml文件然后在里面写入以下代码

<view>

<text class="window">Hello</text>

</view>

然后我们创建index.wxss文件，在文件中输入如下代码

.window{

color: #4994fa;

}

然后我们在创建index.js文件  
在文件中输入如下代码（输入Page IDE会有提示）

Page({

data: {

*//text: "这是一个页面"*

}，

onLoad: function(options) {

*//页面初始化options为页面跳转所带来的参数*

}，

onReady: function() {

*//页面渲染完成*

}，

onShow: function() {

*//页面显示*

}，

onHide：function() {

*//页面隐藏*

}，

onUnload: function() {

*//页面关闭*

}

})

函数解释如下：

**Page**

**Page()函数用来注册一个页面。接受一个Object参数，其制定页面的初始化数据、生命周期函数、事件处理函数等。**

**Object参数说明**

| **属性** | **类型** | **描述** |
| --- | --- | --- |
| data | Object | 页面的初始数据 |
| onLoad | Function | 生命周期函数--监听页面加载 |
| onReady | Function | 生命周期函数--监听页面初次渲染完成 |
| onShow | Function | 生命周期函数--监听页面显示 |
| onHide | Function | 生命周期函数--监听页面隐藏 |
| onUnload | Function | 生命周期函数--监听页面卸载 |
| onPullDownRefeash | Function | 页面相关处理函数--监听页面下拉动作 |
| 其他 | Any | 开发者可以添加任意的函数活数据到Object参数中，用this就可以访问 |

**配置首页**

Json文件负责配置页面的路径  
所以我们在里面可以加入如下代码  
其中index的含义其实就是指index.js文件  
这里需要说明一点pages里面的路径其实是指向js文件的  
如果一个目录下没有该名称的js文件是会报错的！

"pages": [

"pages/index/index"

]

**超级Hello World**

为了学习绑定，以及如何将数据在页面更新  
我们来做一个超级Hwllo World，就是我点击他的时候能让文字变色！

**绑定事件**

我们打开index.wxml将里面的代码改成下面这样

<view>

<text catchtap="click" class="window">Hello</text>

</view>

其实也就是增加了

* catchtap='click'

这两个属性是什么意思呢？慢慢来解释

| **类型** | **触发条件** |
| --- | --- |
| touchstart | 手指触摸 |
| touchmove | 手指触摸后移动 |
| touchcancel | 手指触摸动作被打断，如来电提醒，弹窗 |
| touchend | 手指触摸动作结束 |
| tap | 手指触摸后离开 |
| longtap | 手指触摸后，查过350ms在离开 |

**注：除上表之外的其他组件自定义事件都是非冒泡事件，如<form/>的submit事件，<input/>的input事件，<scroll-view/>的scroll事件。**

**事件绑定**

事件绑定的写法同组件的属性，以key, value的形式。

* key以bind或catch开头， 然后跟上事件的类型，如bindtap, catchouchstart
* value是一个字符串，需要在对应的Page中定义同名的函数。不然当初发事件的时候会报错。

**bind事件绑定不会阻止冒泡事件向上冒泡，catch事件绑定可以阻止冒泡事件向上冒泡。**

上面展示了事件的一些的属性名称，这里需要注意黄色标准的内容，区分冒泡事件和非冒泡事件，其实冒泡事件就是需要往上面一层容器传递这个事件。

看了这段代码我们再来理解catchtap="click"的含义  
catch代表非冒泡事件，tap代表点击事件，所以连在一起就是非冒泡点击事件，后面的click只是一个变量的名字，我们在index.js需要用这个名字绑定接受事件的函数。  
**我们在index.js文件中添加如下函数**

click: function() {

console.log("点击了文字")；

}

添加完成之后index.js中的文件就是如下这样

Page({

data: {

*//text: "这是一个页面"*

},

click: function() {

console.log("点击了文字")

},

onLoad: function(options) {

*//页面初始化options为页面跳转所带来的参数*

console.log("Page OnLoad")

},

onReady: function() {

*//页面显示*

console.log(“Page onReady”);

},

onShow: function() {

*//页面显示*

console.log(“Page onShoe”)

},

onHide: function() {

*//页面隐藏*

console.log("Page onHide")

},

onUnload: function() {

*//页面关闭*

console.log("Page onUnload")

}

})

所以点击事件的回掉函数就是catchtap="click"中的click后面加上：function()构成的，  
当我们点击文字时控制台中打印出了log。

接下来我们来写一下点击变色的逻辑，但是如何让文字变色的呢，肯定是css，所以我们需要在index.wxss中添加一个样式。

.window-red{

color:#D23933;

}

此时index.wxss中的代码为：

.window{

color: #49bb16;

}

.window-red{

color:#D23933;

}

然后我们进入index.js文件，你会发现在代码里面有一个data：{}他不是page生命周期函数，其实他是一个变量的数组，这个里面申请的变量都可以在wxml中使用。

data: {

*//text: "这是一个页面"*

}

我们在这里申请了一个color

data: {

*//text: "这是一个页面"*

color: "window"

}

color的值就是index.wxss的样式名称，然后我们进入index.wxml中,将class中的值改成{{color}}

<view>

<text catchtap='click' class='{{color}}'>Hello</text>

</view>

其实意思就是将js文件中变量color的值在这里使用，也就是值等于window，然后我们在回到index.js文件，在最上面申请一个变量控制点击，然后我们在click：function()函数中添加如下代码

click: function() {

console.log('点击了文字');

if(flag) {

color = "window-red";

flag = false;

} else {

color = "window"

}

this.setData({

color

});

}

其实就是在点击后更换color变量的值，而更换的这个值其实就是样式的名称

**更新界面数据**

这里有一个问题，就是我们在更换完值，但是在wxml中不会立即生效，所以我们需要调用this.setData()方法将值同步给wxml让他立即生效

最后补充一点index目录下也是可以配置json文件的，也就是每个页面都可以配置自己独特的actionbar颜色等等，这里的配置会覆盖app.json文件的配置。