# 1.1 关于app.json

小程序的app.json必须存在，否则无法运行。小程序会将app.json作为配置文件入口，app.json是小程序的全局配置。其中包括了页面的路径，界面表现，网络超时时间，底部的tab等。

{

"pages": [ // 设置页面的路径

"pages/index/index",

"pages/logs/index"

],

"window": { // 用于设置小程序的状态栏，导航条，标题窗口等对象的颜色

"navigationBarTitleText": "Demo"

},

"tabBar": { // 页面底部的菜单，其中list属性，最少支持2个，最多支持5个

// list的item中包含4个属性，路径，文字，图片的路径，和被选中的 图片路径

"list": [{

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

"text": "首页"

}, {

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

"text": "日志"

}]

},

"networkTimeout": { // 用于设置网络请求对象的超时时间。

// 有如下属性：

1. Request 用于wx.request的超时时间
2. connectSocket 用于wx.connectSocket的超时时间
3. uploadFile 用于wx.uploadFile的超时时间
4. downloadFile 用于wx.downloadFile的超时时间

"request": 10000,

"downloadFile": 10000

},

"debug": true,

"navigateToMiniProgramAppIdList": [

"wxe5f52902cf4de896"

]}

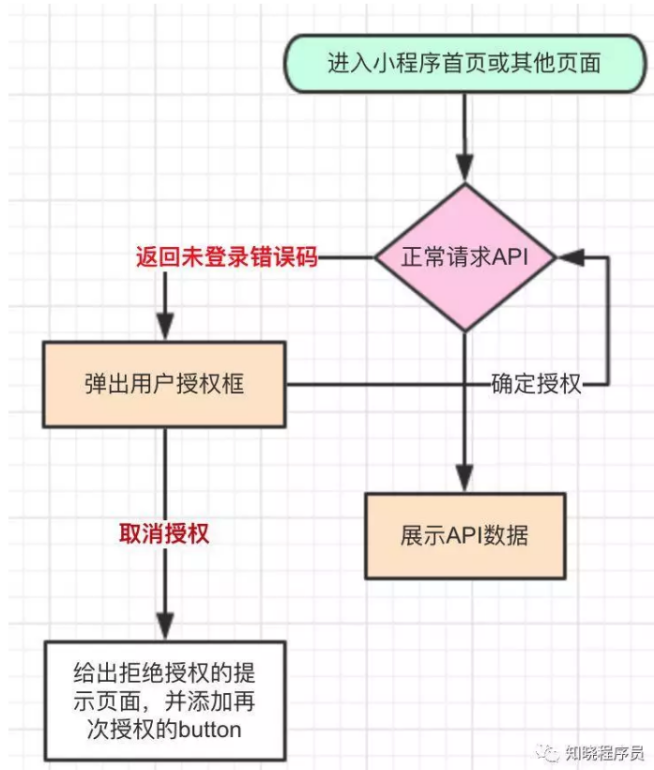
# 1.2 关于app.js

1. 我们不能再app.js中做小程序的登录逻辑

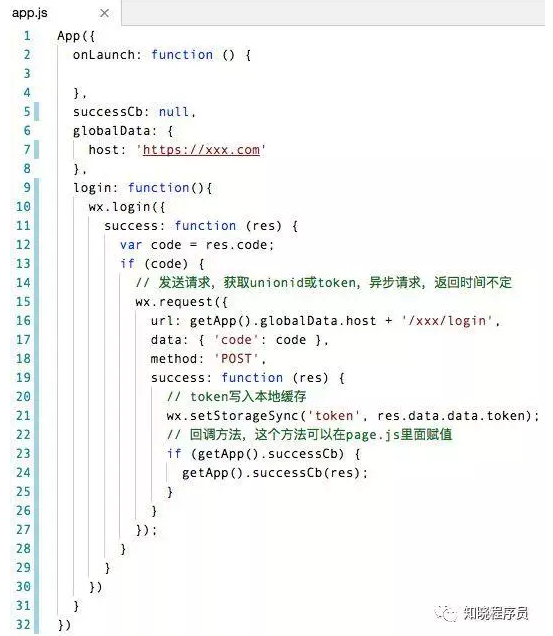
登录的话，势必要使用wx.request，那么就会存在异步，所以api在何时返回数据是无法确定的，如果我们把用户的登录放在app的onLaunch或者onshow里面，然后又在page.js里面用到登录方法返回的结果（结果可能返回用户的头像，名称）。那么就会出现偶尔有头像偶尔没有头像。而且异步的请求只能通过callBack的方式来模拟同步。

所以不建议把登录的方法放在app的生命周期中

正常的登录逻辑应该是这样子的：



1. 微信小程序的登录（wx.login）应该写在哪里？当前有2种解决方案：
2. 写在app.js中，但不是在生命周期里面



但我们在app.js中写完这个login方法以后。在小程序的登录页面，我们可以通过：getApp()来进行获取app的实例，于是就可以app.login来进行调用这个方法

1. 第二种就是将login的方法放在utils中，utils里面存放公共的代码。于是在哪里页面位置，都是是可以调用的。

总结：针对于这两种方法，其实应该更偏向于第二种。

# 1.3 关于project.config.json

没啥用，如果不是个性化的配置，可以忽略

# 1.4 pages里面的文件，分为四种类型

1. wxml文件  
    wxml模板和html非常像，其中你可以把view当做是div，把text当做p标签
2. wxss文件

你可以理解为平时写的css文件，局部的wxss样式文件，仅仅对当前的page页面有效（这一点需要注意！！）如果你想写影响全局的样式，那么需要创建一个wxss样式文件，然后将样式导入到app.wxss文件中。因为只有app.wxss文件才能够影响全局。

同时wxss中新增了一种单位交rpx。（1rpx = 0.5px = 1物理像素）

同时wxss只支持部分css选择器

1. json文件

配置页面中的一些独立属性。

navigationBarTitleText：导航栏标题文字（这个经常会用用到，注意了！！！）

backgroundColor：窗口的背景颜色

onReachBottomDistance：页面上拉触底事件触发时距页面底部距离，单位为px（默认为50）。

disableScroll：设置为 true 则页面整体不能上下滚动。  
只在页面配置中有效，无法在 app.json 中设置

disableSwipeBack：禁止页面右滑手势返回(默认为false)

usingComponents：页面[自定义组件](https://developers.weixin.qq.com/miniprogram/dev/framework/custom-component/)配置（这个经常会用用到，注意了！！！）

1. js文件

# 1.5 小程序的事件

1. 事件的分类

冒泡事件：当一个组件上的事件被触发后，该事件会向父节点传递。

非冒泡事件：当一个组件上的事件被触发后，该事件不会向父节点传递。

1. 事件的绑定

事件的绑定的写法格式以key，value的形式

key可以以bind或者catch开头，然后更上事件的类型，例如：bind:tap

value是一个字符串，需要在对应的page中定对应的方法。

其中bind绑定事件，和catch绑定事件的区别是： 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。

1. 实现下拉刷新，监听用户下拉刷新

需要在app.json的window选项中或者当前的page.json中配置enablePullDownRefresh。

然后可以通过wx.startPullDownRefresh触发下拉刷新，调用后触发下拉刷新动画，效果和用户手动下拉刷新一致。

当处理完数据刷新后，wx.stopPullDownRefresh可以停止当前页面的下拉刷新。

1. 关于事件，有如下：



# 1.6触发事件以后，如何获取对应的参数?(非常重要，请注意！！！)

在wxml中的表现：

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

在js中的表现：

Page({

tapName: function(event) {

const testId = event.currentTarget.dataset.hi;

}

})

事件传递参数方式：在wxml中使用data-参数名=”传递的参数”，注意参数名是不能够大写的，必须小写！！！然后在js中通过：event.currentTarget.dataset.参数名来获取对应的属性。

# 1.7 setData

微信小程序提供了将数据从逻辑层发送到数据层的函数：setData



我们需要注意的是：

1. 直接修改this.data而不是调用this.setData是无法改变页面的状态。
2. 单次设置数据不能超过1024KB，尽量避免一次设置过多的数据
3. 不能频繁调用this.setData，如果是每一帧调用，会造成卡顿

setData的传参：this.setData(object, function)，其中function作为回调函数，你可以理解为vue中的$nextTick。

(1)更新数据

如果我们只想更新data中的某一个数据，某一个属性：

Page({

data: {

User: {

Name: ‘kisure’,

Age: ‘26’

}

},

changeData() {

This.setData({

“user.name”: ‘nice fish’ // 注意了，key必须带引号

})

Const key = “user.name”; // 当然我们也可以将key事先保存

This.setData({ [key]: ‘nice fish’ })

}

})

# 1.8 小程序的路由跳转

1. 路由方式



1. 路由跳转需要注意的点

Navigator，redirectTo只能打开非tabBar页面。

switchTab只能打开tabBar页面。

relaunch可以打开任意的页面，但是没有返回按钮，所以需要定义一个可以点击 回退的按钮。

路由跳转，传递参数以后，我们可以在跳转的page的onLoad的option参数中获取相关的数据。

1. 路由的传参

Wx.navigatoTo({

Url: `/page/orderlist/detail?Id=${id}`

})

// detail页面中：

Page({

Onload(options) {

Console.log(options.id) // 这一步就是获取传递过来的id

}

})

1. 跳转相关的注意事项

如果在跳转页面的时候，存在要重置数据的需求，那么是不能够使用wx.navigateTo(object)的，因为使用navigateTo是不会将原来的页面onUnLoad的(也就是不会卸载原来的页面的，数据还是会保留的)。如果是不属于tabBar中的页面，那么我们可以使用wx.redierctTo(object)来跳转到指定的页面，于是原来的page页面就会被卸载掉。如果是属于tabBar中的页面，使用wx.redirectTo(object)，只能卸载当前的页面，其他的tabBar页面并不会unload(这个是什么操作?我个人认为是，事先点击一个tab，然后再点击第二个tab，而第二个tab中存在一个按钮进行跳转到其他的页面。如果使用wx.redirectTo精心台转，那么只会卸载第二个tab页，而不会卸载第一个tab页)。所以如果希望所有的tab页面都关闭，那么则需要使用wx.relaunch(object)，那么就会卸载之前所有的页面。

其次，navigatorTo是不能跳转到tab页面的。

如果我们想跳转到另一个小程序，那么使用：wx.navigateToMiniProgram({

appId: ‘小程序的appId’,

path: ‘打开的路径页面’,

envVersion: ‘开发的版本’,

success(res) {

// 这个是成功跳转的回调

}

})

# 1.9 关于tabBar的显示问题

在app.json中配置的哪些页面存在于tabBar的list属性中，那么哪些页面就会存在tabBar。否则的话，是不存在底部的tabBar。

小程序中是不存在隐藏tabBar的api。完全依靠跳转来实现。

# 2.0 小程序的生命周期

1. 全局的生命周期

onLaunch(option) {} 初始化完成，全局只执行一次

onShow(){} 启动或者进入前台前都会触发

onHide(){} 小程序进入后台时候触发

执行顺序：onLaunch > onShow

1. page的生命周期

onLoad(option) 页面加载（触发一次），初始化的时候触发，被销毁后进入 触发

onShow() 页面显示，页面进入时候触发

onReady() 页面初次渲染完成(触发完成)，初次渲染完成时候触发，当 前被页面被销毁以后，再次进入该页面渲染完成触发。

onHide() 页面隐藏，进入下一页

onUnload() 页面卸载（也就是被销毁）触发

执行顺序：onLoad > onShow > onReady

1. 组件的生命周期

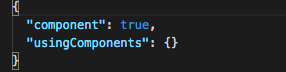
Created 组件的实例刚刚被创建好，于是该生命周期被触发。此时， 组件数据this.data就是组件在构造函数中定义的data。此 时是不能够调用setData的。

Attached 组件完全初始化完毕，进入页面子节点以后，被触发。此时， this.data已被初始化为组件的初始值。该生命周期非常重 要，绝大多数的初始化工作都应该在此处执行。

Detached 组件在离开页面节点数的时候，触发。或者一个page退出 的时候，也会触发组件的生命周期。

# 2.1 自定义组件

编写一个组件，首先，需要在json文件中进行自定义的组件声明：



然后创建js文件：

// 使用component()来注册组件，并提供组件的属性，内部数据和方法

Component({

Properties: {

// 此处定义组件的属性，例如定义一个user属性

User: {

Type: Object,

Value: { // 这个是默认的值

Name: ‘kisure’,

Age: 26

},

},

innerText: {

Type: string,

Value: ‘’

}

},

data: {

// 此处定义组件内部使用的属性

},

Methods: {

// 此处可以定义组件的方法

Callparent() {

//

This.triggerEvent(‘nicefish’, { name: ‘kisure’ });

}

}

})

当我们写完Component以后，就需要在相关的page中使用组件了。

于是我们在先关的page.json中将需要使用的组件，全部声明进去：



此处我们使用的是login.json，在该文件中配置一下。

然后我们在login.wxml中使用：



在wxml中使用的时候，需要记得，如果组件属性存在驼峰，例如：innerText，那么在引用以后写入的属性应该为：inner-text=”....”，其次，组件中有一个this.triggerEvent(‘nicefish’)方法，所以我们可以在wxml中写：bind:nicefish来进行相关的方法绑定触发。

1. 由此，组件的通讯事件是这样子的：

This.triggerEvent(‘事件名字’, object提供给事件监听函数的数据,options触发事件的选项)

Options:

Bubble 默认为false 事件是否冒泡

Composed 默认为false 事件是否可以穿越组件边界，如果是 false,那么事件只能在引用组件的节 点树上触发

capturePhase 默认为false 事件是否拥有捕获阶段

1. 组件的生命周期之前已经提过，可查看2.0【小程序的生命周期】
2. 组件的数据监听器

Component({

....

observers: {

‘data1, data2’: function(param1, param2) {

// 在data1或者data2被设置的时候，就会触发

},

‘user.name’: function(name) {

// 监听属性中的某个字段值变化

// 使用this.setDate() 设置this.data.some.name的时候就会触发

},

‘user.\*\*’: function(fields) {

// 使用setData设置user的任何一个属性，就会触发

// \*\* 这个通配符的意思是监听全部

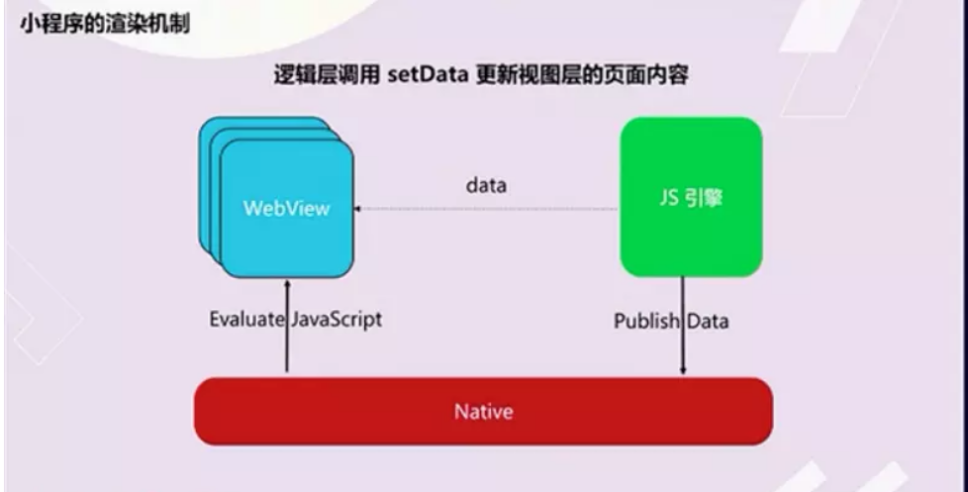
}

}

})

# **2.2 小程序性能优化**

(1) 小程序渲染原理



小程序的逻辑层和渲染层是分开的两个线程。在渲染层，宿主环境会把wxml转化为对应的js对象，在逻辑层发生数据变更的时候，我们需要通过setData方法把数据从逻辑层传递到渲染层。再经过前后的对比差异，把差异应用在原来的Dom树上，然后渲染出正确的UI界面。

总结：页面初始化的时间大致由：页面初始数据通讯时间，初始渲染时间两部分组成。其中，数据通讯的时间指数据从逻辑层开始组织数据到视图层完全接收完毕的时间。传输时间与数据量大体上呈现正相关关系，传输过大的数据将使这一时间显著增加。因此减少传输数据量是降低数据传输时间的有效方式。

1. 影响性能的关键因素

频繁调用setData

每次setData都传递大量新数据

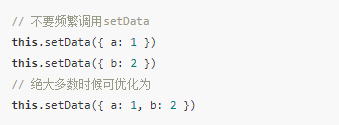
data中放置大量与界面渲染无关的数据

当页面进入后台(用户看不见界面)，则不应该继续进行setData，后台状态时执行setData会抢占前台页面的执行。

1. 优化方案

①减少setData的次数

②合并setData的请求



③去除不必要的事件绑定(wxml中的bind和catch)，从而减少通讯的数据量和次数

④避免在节点的data的前缀属性中绑定过大的数据

⑤列表的局部更新：

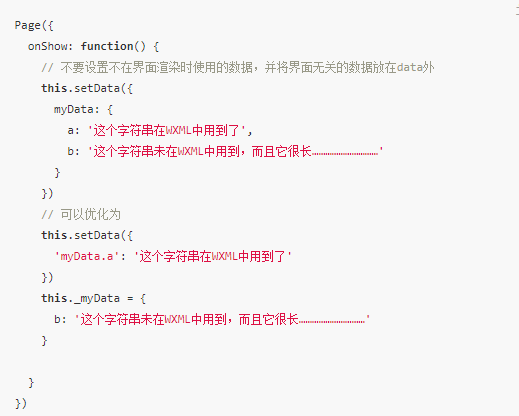
一个list中存在n条数据，当我们为其中一条点赞，那么就使用

This.setData({

‘List[index]’: ......

});

⑥与界面渲染无关的数据不要放在data中



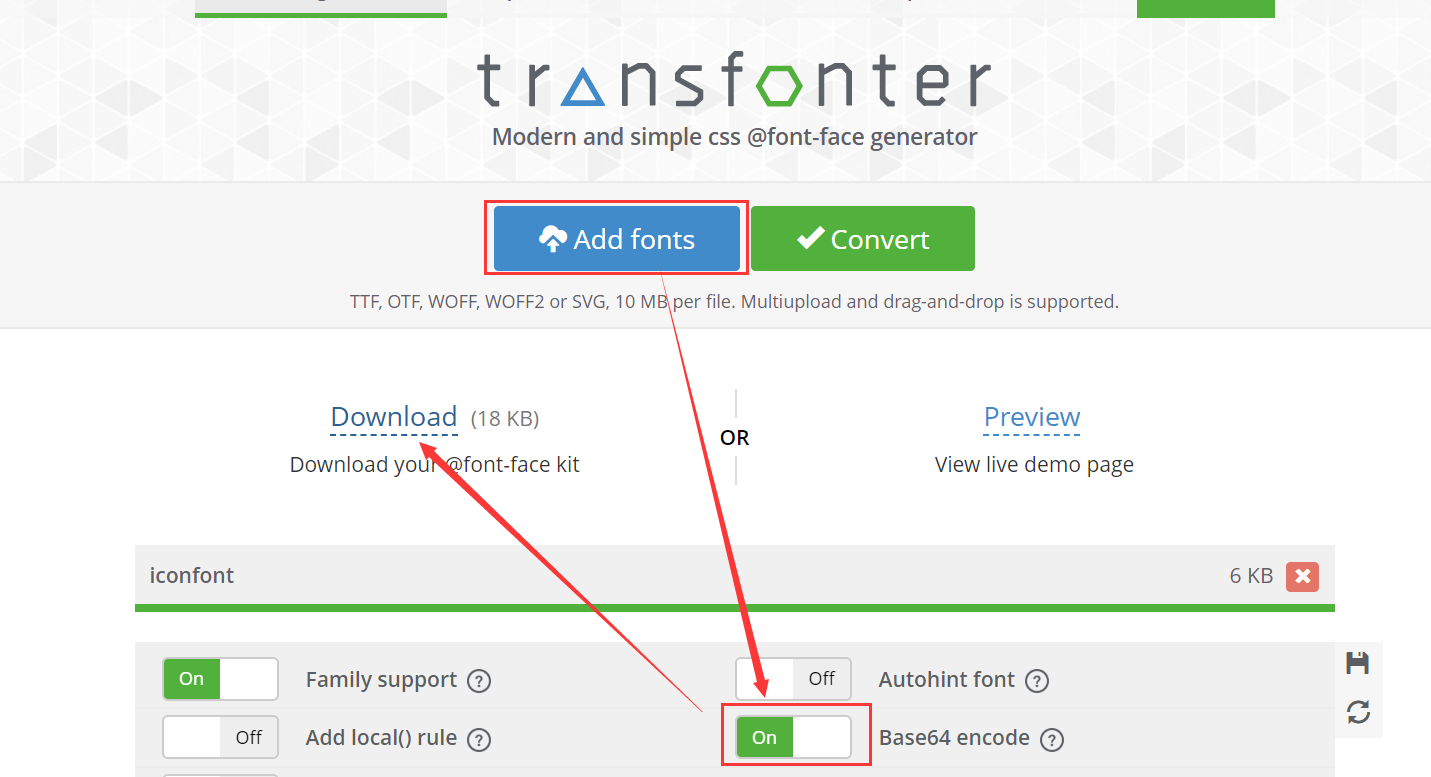
⑦ 防止后台页面的js抢占资源

小程序中存在n个页面，所有的这些页面，虽然都拥有自己的webview，但是却共享一个js运行环境。当跳转到另一个页面，前一个页面如果有定时器之类的，还是会在js环境中执行的。所以需要销毁这类占资源的东西。

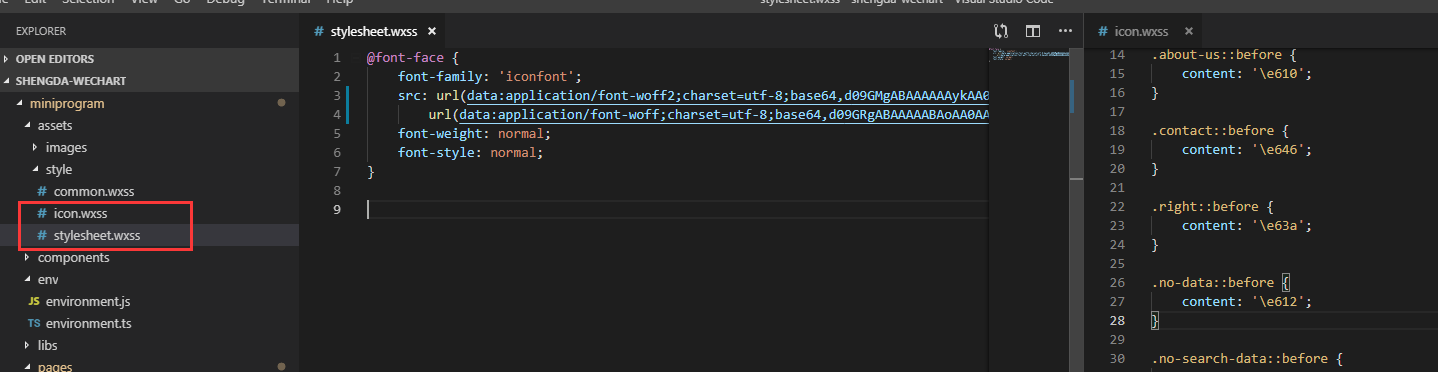
⑧ 谨慎使用onPageScroll

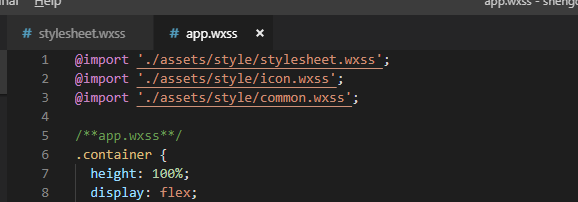
# 2.3 小程序中的字体

目前小程序不支持ttf之类的文件使用。只能通过将ttf文件转化为base64。我们可以打开：transfonter.org，点击Add fonts按钮将字体添加，然后on Base64，转化完以后，直接点击Download下载css文件。



使用：





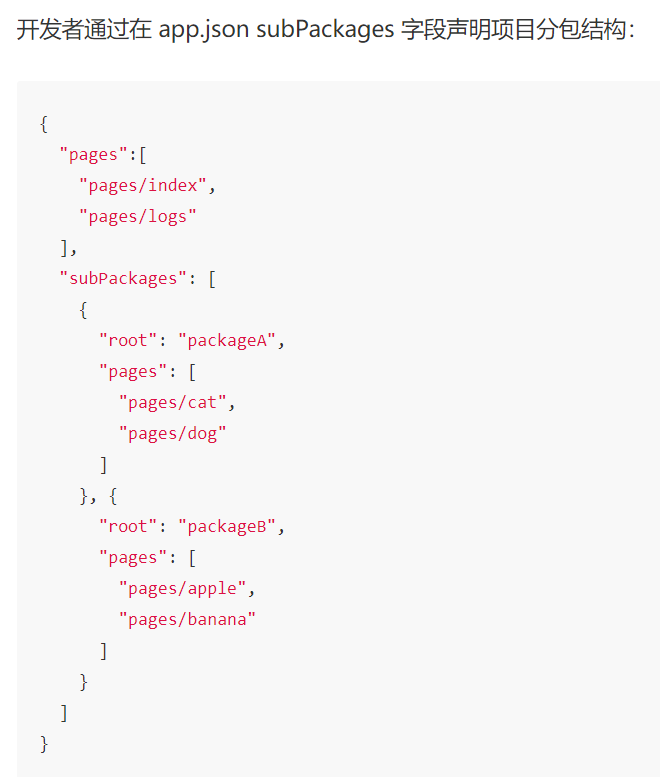
其中stylesheet.css是base64的样式，icon.wxss是自定义的样式，里面的content来自于阿里矢量图16进制的值。然后需要将这两个样式文件导入到app.wxss中。并且在app.wxss中添加样式：

# 2.4 小程序的分包

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要用到公共资源/js脚本。

小程序启动时，默认会下载主包并启动主包内页面，如果用户需要打开分包内某个页面，客户端会把分包下载下来，下载完成后再进行展示。





分包的原则：

1. 声明subPackages后，将按subPackages配置路径进行打包，subPackages配置路径外的目录将被打包到app（主包）中。
2. App（主包）也可以有自己的pages
3. SubPackages的根目录不能是另外一个subPackage内的子目录
4. 首页的tab，必须在app（主包）内

引用的原则：

1. packageA中无法require 包B中的js文件。但是可以require app,自己包中的js

# 2.5 小程序wxml的一些语法

(1) 组件属性：

<view id=”item-{{ id }}”></view>

Page({

data: { id: 0 }

})

(2) Wx:if: 显示隐藏

<view wx:if=”{{show}}”></view>

(3) Wx:for 列表渲染

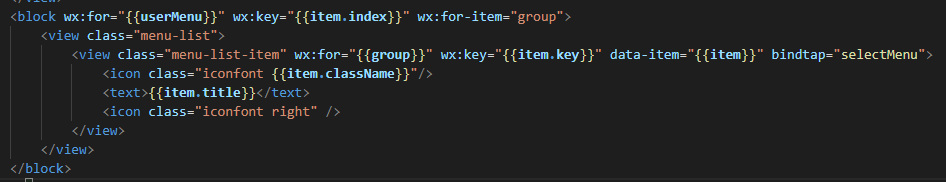
循环显示，你可以理解为v-for或者ng-for或者ng-repeat

<view wx:for=”{{array}}”></view>

使用wx:for-item 指定数组当前元素的变量名

使用wx:for-index 指定数组当前下标的变量名

案例：



(4) block 你可以把它当做vue中的template或者react中的<>/<react.fragment></react.fragment>或者angular2+中的<ng-container>

(5) wx:key 你可以把它当做vue或者react中的key

# 2.6 小程序自定义component中的behavior

behavior是用于组件间代码共享的特性，类似于一些编程语言中的minxins。每个behavior中可以包含一组属性：





# 2.7 http的封装





