1. **小程序组件之间传值和传递样式**
2. 在自定义组件.js文件中使用

properties:{

title:String(数据类型)

}

在其他组件中使用自定义组件的标签中，title=” ”

1. 给title设置个默认值

Properties:{

title:{

Type:String,

value:”我是默认的标题”,

observer:function(newval,oldval){

使用observer监听title值的变化

}

}

}

同样在其他组件中使用自定义的组件标签中，设置title=””,如果没给title赋值的话，会默认显示我是默认的标题

1. 设置样式
2. 在自定义组件中给某个标签设置class属性 例：

<view class=’titleclass’></view>

1. 在自定义组件的.js文件中

externalClasses:[’titleclass’]

1. 在使用自定义的组件的标签中，设置titleclass=’red’ , 在.wxss组件中设置.red的样式
2. **组件之间相互传递事件**

在自定义组件中写入事件时，要写在methods：{}里面，向其他组件传递事件使用：例

html文件中：

<button size='mini' bindtap="handleadd">+1</button>

js文件中

methods:{

handleadd() {

this.triggerEvent('increment',{},{})

}

}

在使用自定义组件的home组件的标签中 例如：<my-event bind:increment=“事件名”> 然后在home组件的js文件中声明事件

1. **小程序的绑定事件传参**

**例如：**<block wx:for="{{titles}}" wx:key="{{index}}">

<view class="tab-item {{currentIndex == index ? 'active' : ''}}" bindtap="clickIndex" data-index="{{index}}"> {{item}}</view>

</block>

在view标签中使用点击事件时想要拿到index,小程序中不能在事件后面加参数，需要用到data,并在js文件中使用如下

clickIndex(**e**) {

this.setData({

currentIndex: e.target.dataset.index

})

}

注：小程序中data参数中的数据重新赋值时，在一个方法中不能直接设置，需要用到this.setData({})如上图，并且在小程序自定义组件中，方法必须写在“methods”中。

1. 父组件向自定义组件传递参数以及传递数据

在自定义组件methods里面自定义一个方法：

methods: {

a(num){

this.setData({

countain: this.data.countain + num

})

}

}

然后在父组件里面：给自定义组件设置class或id属性，在方法里面：

handleupdata(){

const my\_sel = this.selectComponent("#sel-id")

console.log(my\_sel)

my\_sel.a(10)

}

声明自定义组件，并调用自定义组件的方法给其传参

**插槽（slot）**

1. **单个插槽的使用**

自定义组件里面：

<view>我是组件的头部</view>

<slot/>

<view>我是组件的尾部</view>

使用自定义组件的里面：

<my-slot>

我是插入的内容

</my-slot>

1. **多个插槽的使用**

**自定义组件里面wxml文件中**

<view>我是组件的开始</view>

<slot name='slot1'/>

<slot name='slot2'/>

<slot name='slot3'/>

<view>我是组件的尾部</view>

**Js文件中使用**

options:{

multipleSlots:true

},

**在使用自定义组件的组件中**

<my-slot>

<view slot='slot1'>我是插入的内容</view>

</my-slot>

**注：自定义组件中：**

1. **-----让使用者可以给组件传入数据-------**

**Properties:{**

**title:{**

**type：String,**

**Value:‘’**

**}**

**}**

1. **------定义组件内的初始化数据------**

**Data:{**

**Counter:0**

**}**

1. **--------用于定义组件内的函数------**

**methods:{**

**Foo(){**

**}**

**}**

1. **-------定义组件的配置选项-----------**

**//mutipleSlots:在使用多插槽时定义为true**

**//styleIsolation:设置样式的隔离方式**

**Options:{**

**multipleSlots:true**

**}**

1. **--------外界给组件传入额外的样式---------**

**externalClasses:[]**

1. **--------可以监听properties/data的改变--------**

**Oberserver:{**

**Counter:function(newVal){**

**}**

**}**

1. **--------组件中监听生命周期的函数--------**

**监听所在页面的生命周期：**

**pagelifetimes:{**

**Show(){监听所在页面显示出来的**

**}**

**hide(){监听所在页面隐藏时的**

**}**

**resize(){**

**监听页面尺寸的改变**

**}**

**}**

**监听组件本身的生命周期**

**lifetimes:{**

**Created:{**

**组件被创建**

**},**

**attached(){**

**组件被添加到页面中**

**},**

**ready(){**

**组件被渲染出来**

**},**

**movied（）{**

**组件被移动到节点树的另一个位置**

**},**

**detached(){**

**组件被移除到**

**},**

**}**

**小程序系统API**

1. **网络请求 (**<https://developers.weixin.qq.com/miniprogram/dev/api/network/request/wx.request.html>**)**
2. **Get请求：**

**一般在onload（页面加载完成）方法里面使用网络请求:**

**Onload:function(option){**

**wx.request({**

**Url:’http://a.itying.com/api/productlist’，**

**methods：‘get’ methods是请求的方式，**

**Data:{**

**Key:value**

**}**

**Success:(val)=>{ Success请求成功返回的方法里面会有返回的 数据**

**}**

**})**

**}**

**注：当需要get方法传值时，可以在url路径里面拼接，还可以在data里面传值**

1. **post请求**

**Onload:function(option){**

**wx.request({**

**Url:’http://a.itying.com/api/productlist’，**

**methods：‘post’ methods是请求的方式，**

**Data:{**

**Key:value**

**}**

**Success:(val)=>{ Success请求成功返回的方法里面会有返回的 数据**

**},**

fail:(err)=>{ //失败函数

console.log(err)

}

**})**

**}**

1. **封装一个公共的方法进行网络请求**

**首先创建一个js文件使用**

export default function request(option){

return new Promise((resolve,reject)=>{

wx.request({

url: option.url,

method: option.method || 'get',

data: option.data || {},

success: (res) => {

// console.log(res)

resolve(res)

},

fail: (err) => {

reject(err)

// console.log(err)

}

})

})

}

**在使用公共方法的组件的js文件中首先导入方法**

import request from '../service/network.js'

request({

url:'http://a.itying.com/api/productlist'

}).then((res)=>{

console.log(res)

}).catch((err)=>{

console.log(err)

})

并使用这个方法为其赋值

**注：export/import是es6中的语法，export是导出变量和方法供其他组件中使用，其他组件使用时要用import导入变量或方法**

**Promise:**

1. Promise是一个构造函数，所以可以 new 出一个Promise的实例
2. 在Promise上有两个函数 resolve(成功之后的回调函数)和 reject(失败后的回调函数)
3. 在Promise构造函数的prototype属性上，有一个 .then() 方法。所以只要是Promise构造函数创建的实例，都可以访问到 .then()方法
4. Promise表示一个一步操作，每当我们new一个Promise的实例，这个实例就代表具体的异步操作。
5. Promise创建的实例，是一个异步操作，这个异步操作结果，只有两种结果

状态1：异步执行成功，需要在内部调用成功的回调函数resolve把结果返回给调用者

状态2：异步执行失败，需要在内部调用失败的回调函数reject把结果返回调用者

由于Promise的实例是一个异步操作，所以内部拿到操作结果后，无法使用return把操作结果返回给调用者，这个时候只能使用回调函数的形式，把成功或失败的结果，返回给调用者

1. 我们可以在new出来的Promise实例上，调用 .then()方法，预先为这个Promise异步操作，指定成功(resolve)和失败(reject)回调函数

**实现微信分享的两种方式**

1. **在js文件中，使用onShareAppMessage:function(option){**

**Return{**

**Title:’’**

**Path:分享时用户点击进入的页面路径,**

**imageUrl:’’**

**}**

**}**

1. **使用open-type=”share”**

**小程序的登陆流程**

#### 首先是 使用api wx.login 中的success函数获得code，post给服务器

**注：定义全局变量，在app.js中定义变量，在使用全局变量的组件中，使用getApp()方法**

**例：app.js: globalData：{**

**token：’’  
}**

**在使用组件中：const app = getApp()**

**App.globalData.token就是全局变量的值**

**商城源码地址：https://github.com/coderwhy/HYMiniMall.git**