# 基础用法（wxml，js，wcss文件）

写法和普通page相同，使用<slot>节点提供给父节点使用复用，父节点可以给子节点传值。

子节点：

<view class="wrapper">

<view>这里是组件的内部节点</view>

<slot></slot>

</view>

父节点：

<view>

<component-tag-name prop-a="{{dataFieldA}}" prop-b="{{dataFieldB}}">

<!-- 这部分内容将被放置在组件 <slot> 的位置上 -->

<view>这里是插入到组件slot中的内容</view>

</component-tag-name>

</view

1. 通过propA，propB传值，组件中可以使用setData方法来改变绑定得数据字段。绑定得数据这能传递JSON类型得数据，也可以传入函数但这些函数不能在 WXML 中直接调用，只能传递给子组件）。
2. 默认情况下，一个组件只能有一个slot，如果要使用多个slot，需要在js文件中进行声明，不同得slot使用name进行区分

Component({

options: {

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

},

properties: { /\* ... \*/ },

methods: { /\* ... \*/ }

})

# 组件样式：

wcss文件中编写的样式，只对节点wxml文件内部产生作用，使用时需要注意以下几点：

1. 组件和引用组件的界面不能使用id选择器（#a）和属性选择器（[a]），需要使用类选择器。使用后代选择器（.a .b）的时候可能会发生不可预期的结果，尽量少使用。
2. Font，color等样式会从组件外继承到组件内部
3. 除继承样式外， app.wxss 中的样式、组件所在页面的的样式对自定义组件无效（除非更改组件样式隔离选项
4. 组件中可以使用 :host选择器来指定他所在节点的默认样式

通过在js文件的Options中设置styleISolation属性来设置组件的内外影响：

1. isolate（默认）：启用样式隔离，在组件外使用类选择器与组件类选择器重合时不会影响组件的样式。
2. apply-shared：页面的wcss样式将会影响到组件的样式，但自定义组件的样式不会影响页面
3. shared：页面的样式会影响自定义组件，自定义组件wcss定义的样式也会影响页面和其他设置了apply-shared和shared的自定义组件（这个选项在插件中不可以使用）

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

如果组件的Component构造器用于构造页面时，默认值为shared。而且新增了一些新的属性：

1. page-isolated：这个页面禁用app.wcss样式，同时页面的wcss样式不会影响到其他自定义组件
2. page-apply-shared：页面禁用app.wcss样式，页面的wcss样式不会影响其他自定义组件，但是其他设为shared的自定义组件会影响到页面。
3. page-sharde：页面禁用app.wcss样式，会影响其他定义apply-shared和shared的自定义组件，其他定义了shared的自定义组件也会影响它。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

外部样式类：

小程序基础版本库1.9.90开始支持，可以在component中使用externalClasses定义若干个外部样式。



# Component构造器：

Component({

behaviors: [],

properties: {

myProperty: { // 属性名

type: String,

value: ''

},

myProperty2: String // 简化的定义方式

},

data: {}, // 私有数据，可用于模板渲染

lifetimes: {

// 生命周期函数，可以为函数，或一个在methods段中定义的方法名

attached: function () { },

moved: function () { },

detached: function () { },

},

// 生命周期函数，可以为函数，或一个在methods段中定义的方法名

attached: function () { }, // 此处attached的声明会被lifetimes字段中的声明覆盖

ready: function() { },

pageLifetimes: {

// 组件所在页面的生命周期函数

show: function () { },

hide: function () { },

resize: function () { },

},

methods: {

onMyButtonTap: function(){

this.setData({

// 更新属性和数据的方法与更新页面数据的方法类似

})

},

// 内部方法建议以下划线开头

\_myPrivateMethod: function(){

// 这里将 data.A[0].B 设为 'myPrivateData'

this.setData({

'A[0].B': 'myPrivateData'

})

},

\_propertyChange: function(newVal, oldVal) {

}

}

})

# 使用Component构造器构造界面：

小程序的界面也可以被视为自定义组件，因此，页面也可以使用Component构造器构造，拥有与普通组件一样的定义段与实例方法，但是要在对应的json文件中配置“usingComponent”字段

使用组件构造页面后页面也可以接收入参，但要通过跳转的url中带入参数：/pages/index/index?paramA=123&paramB=xyz。 页面的生命周期函数（以on开头的函数）这时要定义在methods字段中。



优点：Component构造器有一个behaviors属性，可以用来提取所有页面中的公用代码段。