# 小程序语法

* 页面文件组成

-pages-



配置内容

-app.json文件-

·”pages”:[ “文件相对路径” , ”文件相对路径” ]

**文件无需写拓展名，默认第一项为首页**

**·**”window”:{ “窗口样式”：样式值, …… } 配置窗口样式，并不是页面的

**·**”debug”: true 开启调试

**·”tabBar” :** {

“color”:：”非选中状态时颜色” ,

“selectedColor”:”被选中时的文字颜色” ,

“backgroundColor”:

“borderStyle”:只能为white或者black,

“position”:”top” 默认”button”,

“list”:[ { “text”:”标签文本内容”,

“pagePath”:”目标页面src”,

“iconPath”:”未选中状态时图标src”,

“selectedIconPath”:”选中状态时图标”,

},

{ }

…] **/\*list成员必须有2个到个\*/**

} 为一个列表对象

**·”networkTimeout”**

-page.jason- 优先于app.json 只能设置window样式

* 逻辑与页面分离结构
* JS的不同

1. 小程序并不是运行在浏览器中，所以没有BOM（window）和DOM（document）对象
2. 一些额外的全局成员：app () {} 用于定义应用程序实例对象

getApp () 用于定义页面对象

Page({ })用于获取全局应用程序对象，没一个页面的逻辑层必须调用该函数来包裹

getCurrentPages 获取当前页面的调用栈，最后一个对象为当前执行页面this

wx.对象 用于提供核心API

1. 支持commonJS规范

* 数据绑定

1. 确认数据在哪 data属性是界面和逻辑间的桥梁

在js中定义 data：{ 变量名：变量值, …… 可以是对象

}

1. 绑定到哪（mustache语法） 在wxml目标位置使用 {{ 变量名 }} 来引用数据
2. 可用于标签内容，或为标签添加属性。例如class=” {{ }} ”
3. 不能用于标签名、属性名
4. {{ }} 内使用 ” ”括住内容，会直接输出括住的字符串
5. 可以在{{ }}中使用表达式，可以直接输出其值
6. 布尔值必须使用”{{ }}”包裹，防止误解

* 界面层的列表渲染

1. wx:for=”{{ 逻辑层中被遍历的目标对象 }}” 例如：<view wx:for=”{{ todos }}”> <><>… </view> 会将view即其包裹的内容按照todos的子内容的个数n循环n次
2. 在界面层中，使用item代替被遍历的目标对象，item . xxx可获取对象的子元素的某个属性值。此处item高于全局的变量item，会导致获取不到全局的item。此时可通过wx:for-item=”XXX”更改item的名字。
3. 界面层中，可通过index获取元素的下标。
4. 在逻辑层，使用 对象名：[ { 属性名1: 值, 属性名2: 值… }, {}… ] 的方式定义一个对象数组

* 界面层的事件处理

1. 语法 界面层的目标元素的属性中添加 事件名=”函数名” 来绑定事件

逻辑层中通过 函数名:function(){ } 来定义事件处理函数函数

1. 相关事件 bindtap catchtap 点击事件

bindinput 表单元素输入事件

bindconfirm 敲回车事件

1. 事件处理函数中的this指向的是当前页面对象Page，并非被点击的元素
2. 普通function中的this指向函数调用者
3. 可通过e.target获取被点击的元素( e即function(e)中的e )

* 事件冒泡

1. bindtap事件会出现冒泡，而catchtap自带阻止事件冒泡的功能

* 界面层事件传参

1. 无法通过 函数名(参数) 直接传参
2. 通过dataset来传参

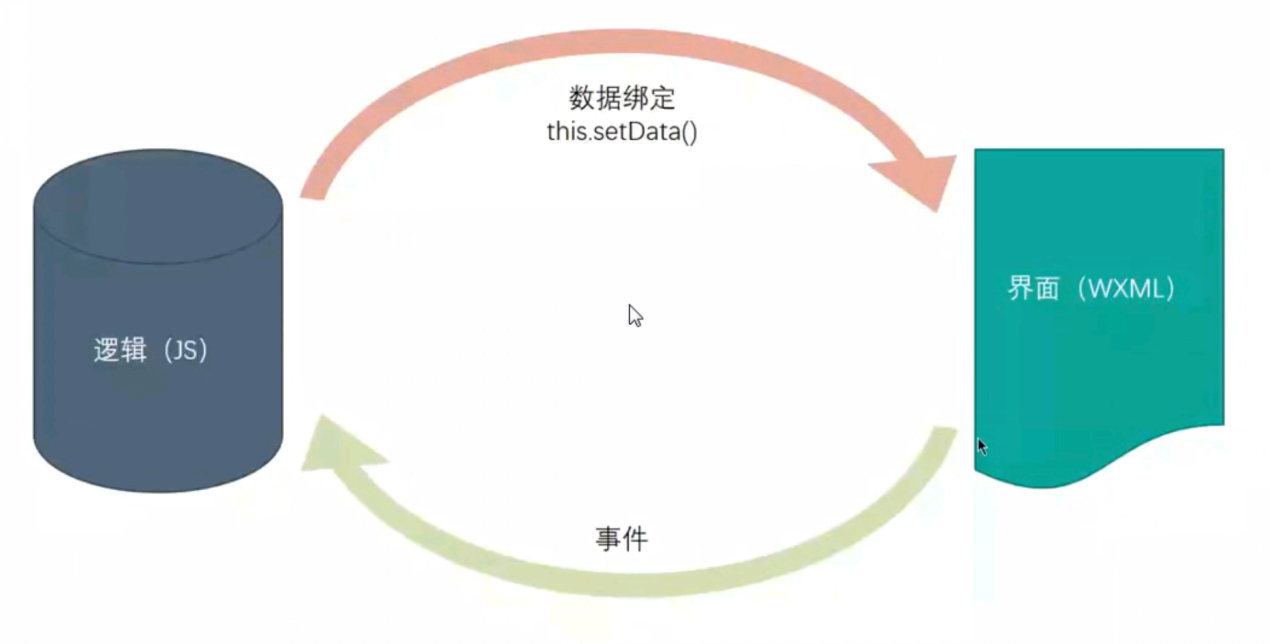
具体做法是 界面层中 使用 **data-自定义属性名=”值”** 写入参数

在逻辑层通过 **e.target.dataset.属性名** 获取参数

自定义属性名不允许有属性名，应该采用 驼峰命名法

1. e.target指向真正被点击的对象，e.currentTarget指向被注册事件的对象(同ES6中的this)

* 数据流

1. 小程序中，数据流都是单向的

2. 小程序中，数据绑定是一次性的，不会随时监视数据的变化

2. 实现双向数据流的方法（即 界面层与逻辑层的数据同步改变）

i．使用**this.setData({ 属性名：目标值 })** 可以实现属性值随着目标值实时变动

ii．input表单中的e.detail.value 为表单中的内容

* 抽象重复的函数

寻找相同部分，将不同的部分设置成变量，采用dataset赋予其不同的变量

* 条件渲染

1. wx:if=”{{ 表达式 }}”，根据表达式的值来决定元素是否显示。
2. wx:elif=”{{ 表达式 }}”，及wx:else 类似。
3. 若有多个同等级view拥有相同的属性，比如wx:if=”{{ !show }}”，可采用<block></block>标签将这些view包裹起来，给block指定该相同属性。注意：该block并不实际存在，**只是一个虚拟的标签，页面渲染过程中不具有实际意义**，仅用于控制多个控制属性( wx:开头的属性 )。
4. 与hidden=”{{ 表达式 }}”的区别：hidden不能作用于block等虚拟属性；wx:if是惰性的，当值为假时，不被渲染

* WXSS与CSS的差异

1. rpx：750rpx占据整个屏幕的宽度
2. rem：规定20rem占据整个屏幕的宽度
3. 选择器

* User Interface（UI）组件

1. 组件是视图层的一系列基础组件
2. 基础内容
3. icon图标 属性：type用于定义规定范围内的图标类型。其他图标必须通过图片形式导入
4. 表单组件
5. 语法问题，类似于HTML，使用严格的XML标准

* 布局

1. 视图容器view标签

* 页面间跳转

1. 创建多个界面技巧：在app.json中的pages栏写入想要新建页面的相对地址src
2. json文件若无内容，必须删除。或者可以加入空括号 {} 来避免出错
3. navigator超链接标签： i. url = ”目标页面相对路径”

ii. redirect= ”{{ true }}” 跳转到目标页面后，取消返回按钮

iii． hover-class= “ 类名 ” 链接被点击后的类名，若类名为none，则为无样式

1. 所有元素均可使用courser: pointer 样式为图片添加点击高亮

* 页面间传值

1. 发送方式 url = ” 目标页面相对地址**?name=值** ”
2. 获取方式 目标页面中的逻辑层 页面初始化函数

onload : function ( **options** ){ options即为传递的参数 }

* 页面导航API

1. wx.navigatorTo({

url:’ ’,

success:function(){回调函数},

fail(){回调函数},

complete:function(){回调函数}

})

1. wx.redirectTo({ url=” ” }) 相当于redirect
2. wx.navigateBack({ delta:**n** }) 默认返回上一页。n为返回的层数，当超出历史记录时，返回首页

* 数组

数组.push({ }) 往数组中添加一个对象

移除元素 var item=对象.splice(目标元素的起始下标，移除的个数) item为被删除的元素组成的**数组**

遍历方法 数组**.**forEach ( function ( 形参代表子元素 ){回调函数} ) 数组为空时，元素不会被传递进回调函数

过滤遍历 数组**.**filter( function( 形参代表子元素 ) { return 过滤条件 } )

**// 未确定** 遍历创建新数组 旧数组.map( function () )

* 小程序发送HTTP请求

1. API wx.request({

url:’ ’ ,

success **:** function( res ){ }

})

