WXML

一、WXML模块语法---数据绑定

1、数据绑定的基本原则

```
    1.在data中定义数据

    2.在WXML中使用数据
```

2、在data中定义页面的数据

在页面对应的.js文件中，把数据定义到data对象中即可：

```
Page({
   data:{
    // 字符串类型的数据
    info:'init data',
    // 数组类型的数据
    msgList:[{msg:'hello'},{msg:'world'}]
  }
})
```

3、Mustache语法的格式

把data中的数据绑定到页面中渲染，使用Mustache语法（双大括号）将变量抱起来即可。

```
<view>{{要绑定的数据名称}}</view>
```

4、Mustache语法的应用场景

```
Page({
   data:{
      imgSrc:'图片路径'
  }
})
```

页面的结构如下:

```
<image src="{{imgSrc}}"></image>
//如果不希望图片变形，可在image后加入mode=""属性
```

例子2

```
<view> {{ message }} </view>
```

```
Page({
  data: {
    message: 'Hello MINA!'
  }
})
```

5.运算（三元运算、算数运算等）

可以在 {{}} 内进行简单的运算，支持的有如下几种方式:

三元运算

```
<view hidden="{{flag ? true : false}}"> Hidden </view>
```

算数运算

```
<view> {{a + b}} + {{c}} + d </view>
Page({
  data: {
    a: 1,
    b: 2,
    c: 3
  }
})
```

view中的内容为 3 + 3 + d

数据路径运算

```
<view>{{object.key}} {{array[0]}}</view>
```

```
Page({
  data: {
    object: {
      key: 'Hello '
    },
    array: ['MINA']
  }
})
```

字符串运算

```
<view>{{"hello" + name}}</view>
```

```
Page({
  data:{
    name: 'MINA'
  }
})
```

（三元运算） 页面的数据如下：

```
Page({
   data:{
     randomNum:Math.random()*10 //生成10以内的随机数
  }
})
```

页面的结构如下：

```
<view>{{randomNumber>=5?'随机数大于或等于5':'随机数字小于5'}}</view>
//如果希望查看当前所有数据，可以通过调试器里的AppData查看所有数据
```

(算数运算）页面的数据如下：

```
Page({
   data:{
    randomNum:Math.random(),toFixed(2) //生成一个带两位小数的随机数，例如0.34
  }
})
```

页面的结构如下：

```
<view>生成100以内的随机数：{{randomNum*100}}</view>
```

二、WXML模块语法---事件绑定

1.小程序中常用的事件：

类型	绑定方式	事件描述
tap	bindtap 或 bind:tap	手指触摸后马上离开，类似于 HTML 中的 click 事件
input	bindinput 或 bind:input	文本框的输入事件
change	bindchange 或 bind:change	状态改变时触发

2.在小程序中，不存在HTML中的onclick鼠标点击事件，而是通过tap事件来响应用户的触摸行为。

1.通过bindtap，可以为组件绑定tap触摸事件，语法如下

```
按钮
```

2.在页面的.js文件中定义对应的事件处理函数，事件参数通过形参event(一般简写成e）来接收：

```
Page({
   data:{
   },
   btnTapHandler(e){ //按钮的tap事件处理函数
     console.log(e) //事件参数对象e
  }
})
```

3.在事件处理函数中为data中的数据赋值

通过调用this.setData(dataObjeck)方法，可以给页面data中的数据重新赋值，示例如下：

```
+1
```

```
//页面的.js文件
Page({
   data:{
      count:0
   },
  //修改count的值
   changCount(){
    this.setData({
      count:this.data.count+1
     })
    }
  }
})
```

4.事件传参

> 注：小程序中的事件传参比较特殊，不能在绑定事件的同时为事件处理函数传递参数。
>
> 可以为组件提供 data-*自定义属性传参，其中*代表的是参数的名字，示例代码如下：

事件传参

info会被解析为参数的名字、数值2会被解析为参数的值

在事件处理函数中，通过event.target.dataset.参数名即可获取到具体参数的值，示例代码如下：

```
btnHandler(event){
    //dataset是一种对象，包含了所有通过data-*传递过来的参数值
    console.log(event.target.dataset)
    //通过dataset可以访问到具体参数的值
    console.log(event.target.dataset.info)
}
```

5.bindinput的语法格式

在小程序中，通过input事件来响应文本框的输入事件，语法格式如下：

1.通过bindinput，可以为文本框绑定输入事件：

```
<input bindinput=""inputHandler"></input>
```

2.在页面的.js文件中定义事件处理函数：

```
inputHandler(e){
//e.detail.value 是变化过后，文本框最新的值
console.log(e.detail.value)
}
```

6.实现文本框和data之间的数据同步

步骤：1.定义数据

```
Page({
    data:{
        msg:'你好'
      }
   })
```

2.渲染结构

```
{{msg}}
```

3.美化样式

```
input{
border:1px solid #ccc;
padding:5px;
margin:5px;
border-radius:3px;
}
```

4.绑定input事件处理函数

```
//文本框内容改变的事件
iptHandler(e){
    this.setData({
        //通过e.detail.value获取到文本框最新的值
        msg:e.detail.value
    })
}
```

三、WXML模板语法---条件渲染

1、wx:if

在小程序中，使用wx:if="f{condition}}"来判断是否需要渲染该代码块:

```
<view wx:if="{{condition}}">True</view>
            也可以用wx:elif 和 wx:else来添加else判断:

<view wx:if="{{type===1}}">男</view>
<view wx:elif="{{type===2}}">女</view>
<view wx:else>保密</view>
<view wx:if="{{length > 5}}"> </view>
```

2、结合使用wx:if

如果要一次性控制多个组件的展示与隐藏，可以使用一个标签将多个组件包装起来，并在标签上使用wx:if 控制属性，示例如下:

```
<block wx:if="{{true}}">
    <view>view1</view>
    <view>view2</view>
</block>
```

注意: 并不是一个组件，它只是一个包裹性质的容器，不会在页面中做任何渲染。

3、hidden

在小程序中，直接使用hidden="{f condition }"也能控制元素的显示与隐藏:

```
<view hidden="{{condition}}">条件为true隐藏，条件为false显示</view>
```

4、wx:if与hidden的对比

> *1.运行方式不同*
> *wx:if以动态创建和移除元素的方式，控制元素的展示与隐藏*
>
> *hidden以切换样式的方式(display: none/block;)，控制元素的显示与隐藏*
> *2.使用建议*
> *控制条件复杂时，建议使用wx:if 搭配wx:elif、wx:else进行展示与隐藏的切换*

四、WXML模板语法---列表渲染

1、wx:for

通过wx:for可以根据指定的数组，循环渲染重复的组件结构，语法示例如下:

```
<view wx:for="{{array}}">
索引是: {{index}}当前项是: {{item}}
</view>
```

2、wx:key的使用

类似于Vue列表渲染中的:key，小程序在实现列表渲染时，也建议为渲染出来的列表项指定唯一的 key值,从而提高渲染的效率，示例代码如下:

```
//data数据
data{
    userList:[
    {id:1,name:'小明'},
    {id:2,name:'小红'},
    {id:3,name:'小蓝'},
]
}
```

```
<view wx:for="{{userList}}" wx:key="id">{{item.name}}</view>
```

五、属性

组件属性(需要在双引号之内)

```
<view id="item-{{id}}"> </view>
Page({
  data: {
    id: 0
  }
})
```

控制属性(需要在双引号之内)

```
<view wx:if="{{condition}}"> </view>
Page({
  data: {
    condition: true
  }
})
```

关键字(需要在双引号之内)

`true`：boolean 类型的 true，代表真值。

`false`：boolean 类型的 false，代表假值。

```
<checkbox checked="{{false}}"> </checkbox>
```

特别注意：不要直接写 `checked="false"`，其计算结果是一个字符串，转成 boolean 类型后代表真值。