视图类容器组件.

① view (div).

② scroll-view (可滚动列表).

③ swiper 轮播图容器

swiper-item. 轮播图 item. 组件.

swiper 组件常用属性

属性	类型	默认值.	说明
indicator-dots	boolean	false	是否显示指示点
indicator-color	color	rgba.	指示点颜色
indicator-active-color	color	#000000	当前指示点颜色
autoplay	boolean	false	是否自动切换
interval	number	5000ms	切换时间间隔
circular	boolean	false.	是否衔接滑动

text 组件.

text中的 selectable 属性，实现长按选中文本内容.

例：<view>

XXXXXX

<text selectable>133 XXXX</text>.

</view>.

注：133XXXX. 支持长按选中. XXXXXX不支持.

rich-text：

通过 rich-text 组件的 nodes 属性节点，把 HTML 字符串渲染为对应 UI.

例：<rich-text nodes="<h1 style='color:red;'>标题</h1>"></rich-text>

Images 属性与 html. 类似.（注.有默认值.约.300px，250px）.

Image 中的 mode 属性.如下.

mode 值	说明
scaleToFill	（默认值）缩放模式，不保持纵横比缩放图片，使图片的宽高完全拉伸至填满 image 元素
aspectFit	缩放模式，保持纵横比缩放图片，使图片的长边能完全显示出来。也就是说，可以完整地将图片显示出来。
aspectFill	缩放模式，保持纵横比缩放图片，只保证图片的短边能完全显示出来。也就是说，图片通常只在水平或垂直方向是完整的，另一个方向将会发生截取。
widthFix	缩放模式，宽度不变，高度自动变化，保持原图宽高比不变
heightFix	缩放模式，高度不变，宽度自动变化，保持原图宽高比不变

小程序官方把 API 分为了如下 3 大类：

① 事件监听 API

- 特点：以 on 开头，用来监听某些事件的触发
- 举例：wx.onWindowResize(function callback) 监听窗口尺寸变化的事件

② 同步 API

- 特点1：以 Sync 结尾的 API 都是同步 API
- 特点2：同步 API 的执行结果，可以通过函数返回值直接获取，如果执行出错会抛出异常
- 举例：wx.setStorageSync('key', 'value') 向本地存储中写入内容

③ 异步 API

- 特点：类似于 jQuery 中的 $.ajax(options) 函数，需要通过 success、fail、complete 接收调用的结果
- 举例：wx.request() 发起网络数据请求，通过 success 回调函数接收数据

数据绑定.

声明数据：

.js 文件中的. data:

例. data:{ info:'Hello World'},

调用数据：

<view>{{ info }}</view>.

即可显示 Hello World .

动态绑定属性

声明：

data:{

imgSrc:'http://www.~'

}

调用：

<image src="{{imgSrc}}"></image>.

算术运算.

```
Page({
  data:{
    randomNum: Math.random().toFixed(2).
  }
})
```

生成一个两位小数的随机数

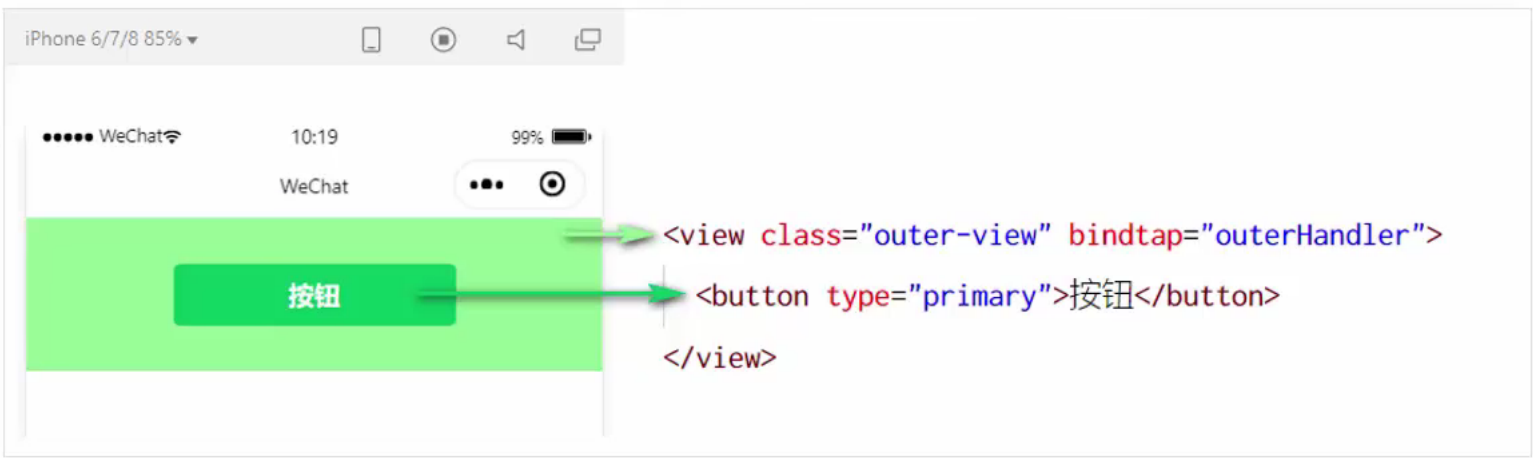
事件：

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

事件对象 event.

属性	类型	说明
type	String	事件类型
timeStamp	Integer	页面打开到触发事件所经过的毫秒数
target	Object	触发事件的组件的一些属性值集合
currentTarget	Object	当前组件的一些属性值集合
detail	Object	额外的信息
touches	Array	触摸事件，当前停留在屏幕中的触摸点信息的数组
changedTouches	Array	触摸事件，当前变化的触摸点信息的数组

target 是触发该事件的源头组件，而 currentTarget 则是当前事件所绑定的组件。举例如下：



点击内部的按钮时，点击事件以冒泡的方式向外扩散，也会触发外层 view 的 tap 事件处理函数。

此时，对于外层的 view 来说：

- e.target 指向的是触发事件的源头组件，因此，e.target 是内部的按钮组件
- e.currentTarget 指向的是当前正在触发事件的那个组件，因此，e.currentTarget 是当前的 view 组件

事件对象：

<button type="primary" data-infor="{{2}}">+2</button>

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 是一个对象，包含了所有通过 data-* 传递过来的参数项
  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)
}
```

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)
}
```

实现文本框和 data 之间的数据同步，

实现步骤：

1. 定义数据.

② 渲染结构.

③ 美化样式.

④ 绑定 input 事件处理函数.

① 定义数据.

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

② 渲染结构.

```
<input value="{{msg}}" bindinput="xxxx"></input>
```

③ 美化样式.

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

④ .js 中绑定 input 事件处理函数.

aqua = #00ffff 即.浅绿色.

```
xxxx(e){
    this.setData({
        msg:e.detail.value //通过 e.detail.value 获取到文本框最新值.
    })
}
```

条件渲染.

1. wx:if

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

```
1 <view wx:if="{{condition}}"> True </view>
```

也可以用 wx:elif 和 wx:else 来添加 else 判断：

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

2. 结合 <block> 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 <block></block> 标签将多个组件包装起来，并在 <block> 标签上使用 wx:if 控制属性，示例如下：

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

注意： <block> 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="{{ condition }}" 也能控制元素的显示与隐藏：

```
1 <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

- 频繁切换时，建议使用 hidden
- 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

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：

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

默认情况下，当前循环项的索引用 index 表示；当前循环项用 item 表示。

数组的定义：array = ['苹果', '华为', '小米']

3. 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：

```
userlist 即数组名.
```

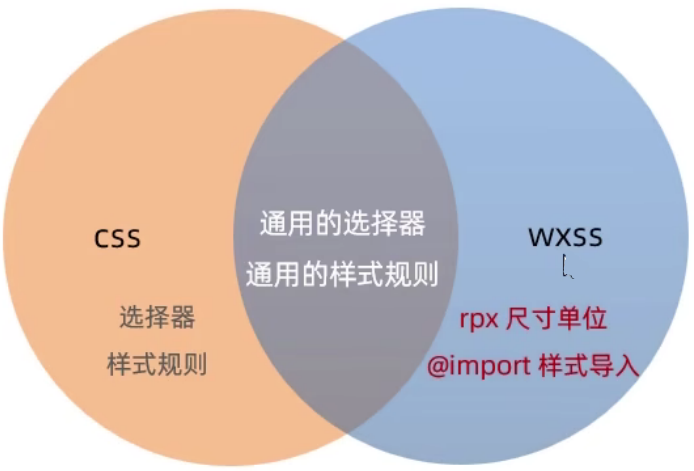
```
// data 数据
data: {
  userList: [
    { id: 1, name: '小红' },
    { id: 2, name: '小黄' },
    { id: 3, name: '小白' }
  ]
}

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

WXSS 和 CSS 的关系.

WXSS 具有 CSS 大部分特性，同时，WXSS 还对 CSS 进行了扩充以及修改，以适应微信小程序的开发。

与 CSS 相比，WXSS 扩展的特性有：

- rpx 尺寸单位
- @import 样式导入



rpx：(responsive pixel) 用来解决屏幕适配的尺寸单位.

rpx 原理：在屏幕宽度上等分为750份