事件

绑定事件：

在组件上使用`bind:事件名称="执行的函数"`即可绑定事件。然后可以在`js`文件中，实现具体的`执行的函数`。以后在这个组件上发生了`bind`后的事件，那么就会触发`js`中的函数了。示例代码如下：

```
<view bind:tap='onViewClick'>
点击我
</view>
```

在js代码中：

```
Page({
    // 前面的代码
    ...
    onViewClick: function(event){
        console.log('view被点击了！ ');
    }
});
```

获取元素上的数据：

我们能捕获到点击事件还不够，我们在点击的时候还要获取一些数据。比如文章列表，我点击了某个文章列表的容器，我想获取这个容器的对应的文章，那么就需要把这个文章的id绑定到`view`上面，以后我在点击的时候，这个数据再通过`event`参数传递给后台的`js`函数。绑定数据的时候，我们需要通过`data-数据名`的方式绑定。示例代码如下：

```
<view wx:for="" data-id="" bind:tap='onArticleClick'>

</view>
```

`js`代码如下：

```
Page({
    onArticleClick: function(event){
        console.log(event);
    }
});
```

事件冒泡和事件捕获：

绝大部分小程序定义好的事件都是冒泡的。冒泡是什么意思呢，就是点击一个子元素，如果事件是冒泡的，那么这个事件也会传递给父元素。如在下边这个例子中，点击`inner view`会先后调用`handleTap2`和`handleTap1`。

```
<view id="outer" bindtap="handleTap1">
  outer view
  <view id="inner" bindtap="handleTap2">
      inner view
  </view>
</view>
```

如果我们想要阻止事件的冒泡。我们可以以 `catch` 开头来定义一个事件。这样就可以拦截事件的冒泡了。如在下边这个例子中，点击 `inner view` 会先后调用 `handleTap3` 和 `handleTap2` (因为tap事件会冒泡到 `middle view`，而 `middle view` 阻止了 `tap` 事件冒泡，不再向父节点传递)，点击 `middle view` 会触发 `handleTap2`，点击 `outer view` 会触发 `handleTap1`。

```
<view id="outer" bindtap="handleTap1">
  outer view
  <view id="middle" catchtap="handleTap2">
    middle view
    <view id="inner" bindtap="handleTap3">
      inner view
    </view>
  </view>
</view>
```

target和currentTarget区别：

```
<view id="outer" bindtap="handleTap1">
  outer view
  <view id="inner" bindtap="handleTap2">
      inner view
  </view>
</view>
```

拿这个例子来讲，如果点击了 `inner view`，那么在 `handleTap2` 中 `target` 和 `currentTarget` 指向的都是同一个对象，而在 `handleTap1` 中，则 `target` 指向的是 `inner view`，`currentTarget` 指向的是 `outer view`。