疯狂早茶微信小程序基础篇《七》：滑动操作

在实际的移动应用程序交互方式中，最常见的就是滑动操作。像左右滑动切换页面，手指张开来放大图片等，都是由滑动操作来完成的。

微信小程序默认提供的相关事件如下：

类型	触发条件
touchstart	手指触摸动作开始
touchmove	手指触摸后移动
touchcancel	手指触摸动作被打断，如来电提醒，弹窗
touchend	手指触摸动作结束
tap	手指触摸后马上离开
longtap	手指触摸后，超过350ms再离开

tap 对应点击操作，还提供了 **longtap** 来支持长按操作，这些都比较简单，就不多做讲述。

touchmove 对应滑动操作，通过 `bindtouchmove` 即可响应滑动操作。

```
//wxml
<view id="id" bindtouchmove="handletouchmove" style = "width : 100px; height : 100px; background : #167567;">
</view>


//js
Page({
   handletouchmove: function(event) {
      console.log(event)
   },
})
```

当按住 view 标签并滑动鼠标时，会不停的触发滑动事件，直到放开鼠标，**当鼠标移出 view 标签区域后依然会触发事件。**

拖拽操作

通过监听滑动事件，可以实现一些实用的功能，比如用过 iphone 的用户都知道 assistivetouch，一个桌面上的快捷按钮，可以将按钮拖动到桌面的任意位置。为了方便，这里就用一个圆形来代表该按钮。

```
//wxml
```

```
<view id="id" class = "ball" bindtouchmove="handletouchmove" style = "width : 60px; height : 60px; background :
#545345;">
</view>

//wxss
.ball {
    box-shadow:2px 2px 10px #AAA;
    border-radius: 20px;
    position: absolute;
}

//js
Page({
    handletouchmove: function(event) {
        console.log(event)
    },
```

})

好吧，按钮丑了点，这不是重点。拖拽操作的实现思路也很简单，当触发滑动事件时，event 对象会包含当前触摸位置的坐标信息，可以通过 event.touches[0].pageX 和 event.touches[0].pageY 来获取，为什么 touches 是数组呢，答案是为了支持多点触控(在电脑上不知道如何模拟多点触控)。接下来将按钮的位置设置为触摸位置，应该就能实现预期效果了，让我们试试看。

在 Page 中定义按钮的位置数据 ballBottom 和 ballRight，并绑定到 view 的对应属性中。

//wxml

<view id="id" class = "ball" bindtouchmove="handletouchmove" style = "width : 60px; height : 60px; background : #545345;

top:{{ballTop}}px; left: {{ballLeft}}px">

</view>

//js

```
Page({
    data: {
        ballTop: 0,
        ballLeft: 0,
    },
    handletouchmove: function(event) {
        console.log(event)
    },
})
```

接下来在 handletouchmove 方法中更新按钮的位置数据

```
handletouchmove: function(event) {
    console.log(event)
    this.setData ({
        ballTop: event.touches[0].pageY,
        ballLeft: event.touches[0].pageX,
    });
```

},

运行发现基本可以实现效果，不过有两个问题，一是会将按钮拖出屏幕边缘，二是按钮始终在鼠标右下方。

接下来加入对屏幕边界的判断并对按钮位置进行修正。其中屏幕大小可以通过 wx.getSystemInfo 来获取。

完整代码如下：

```
Page({
  data: {
    ballTop: 0,
    ballLeft: 0,
    screenHeight:0,
    screenWidth:0

  },
  onLoad: function () {
      //获取屏幕宽高
    var _this = this;
    wx.getSystemInfo({
```

```
      success: function (res) {
          _this.setData({
            screenHeight: res.windowHeight,
            screenWidth: res.windowWidth,
          });
        }
    });
},
handletouchmove: function(event) {
    console.log(event)
    let pageX = event.touches[0].pageX;
    let pageY = event.touches[0].pageY;
    //屏幕边界判断
    if (pageX < 30 || pageY < 30)
      return;
    if (pageX > this.data.screenWidth - 30)
```

```
            return;
        if (pageY > this.data.screenHeight - 30)
            return;
        this.setData ({
            ballTop: event.touches[0].pageY - 30,
            ballLeft: event.touches[0].pageX - 30,
        });
    },
})
```

再次运行，一切 ok。

手势识别

通过处理滑动操作可以识别各种手势操作，如左右滑动等。思路也很简单，通过绑定 touchstart 和 touchmove 事件，然后对坐标信息进行识别计算即可(如 current.PageX - last.PageX < 0 则认为是向左滑动)

```
//wxml
<view id="id" class = "ball" bindtap = "handletap" bindtouchstart = "handletouchtart" bindtouchmove="handletouchmove"
style = "width : 100%px; height : 40px;">
```

```
{{text}}
</view>

//js
Page({
  data: {
    lastX: 0,
    lastY: 0,
    text : "没有滑动",
  },
  handletouchmove: function(event) {
    console.log(event)
    let currentX = event.touches[0].pageX
    let currentY = event.touches[0].pageY

    console.log(currentX)
```

```
    console.log(this.data.lastX)
    let text = ""
    if ((currentX - this.data.lastX) < 0)
      text = "向左滑动"
    else if (((currentX - this.data.lastX) > 0))
      text = "向右滑动"

    //将当前坐标进行保存以进行下一次计算
    this.data.lastX = currentX
    this.data.lastY = currentY
    this.setData({
      text : text,
    });
},

handletouchtart:function(event) {
```

```
    console.log(event)
    this.data.lastX = event.touches[0].pageX
    this.data.lastY = event.touches[0].pageY
  },
  handletap:function(event) {
    console.log(event)
  },
})
```

运行程序，当向左滑动时会 view 会显示"向左滑动"，向右同理。

同时识别左右滑动和上下互动

有时候希望同时识别左右和上下滑动，可以通过比较 X 轴上的差值和 Y 轴上的差值，较大的差值为滑动方向。

```
 handletouchmove: function(event) {
    console.log(event)
    let currentX = event.touches[0].pageX
    let currentY = event.touches[0].pageY
    let tx = currentX - this.data.lastX
```

```
let ty = currentY - this.data.lastY
let text = ""
//左右方向滑动
if (Math.abs(tx) > Math.abs(ty)) {
    if (tx < 0)
        text = "向左滑动"
    else if (tx > 0)
        text = "向右滑动"
}
//上下方向滑动
else {
    if (ty < 0)
        text = "向上滑动"
    else if (ty > 0)
        text = "向下滑动"
}
```

```
    //将当前坐标进行保存以进行下一次计算
    this.data.lastX = currentX
    this.data.lastY = currentY
    this.setData({
      text : text,
    });
  },
```

在实际应用中，当某种手势被触发后，在用户没有放开鼠标或手指前，会一直识别为该手势。比如当用户触发左滑手势后，这时再向下滑动，仍要按照左滑手势来处理。

可以定义一个标记来记录第一次识别到的手势，如果已识别出手势，后续的滑动操作就可以忽略，直到用户放开鼠标或手指。放开鼠标或手指操作可以通过绑定 handletouchend 事件来处理。

```
Page({
  data: {
    lastX: 0,
    lastY: 0,
```

```
    text : "没有滑动",
    currentGesture: 0,
},
handletouchmove: function(event) {
    console.log(event)
    if (this.data.currentGesture != 0){
        return
    }
    let currentX = event.touches[0].pageX
    let currentY = event.touches[0].pageY
    let tx = currentX - this.data.lastX
    let ty = currentY - this.data.lastY
    let text = ""
    //左右方向滑动
    if (Math.abs(tx) > Math.abs(ty)) {
        if (tx < 0) {
```

```
            text = "向左滑动"
            this.data.currentGesture = 1
        }
        else if (tx > 0) {
            text = "向右滑动"
            this.data.currentGesture = 2
        }

    }
    //上下方向滑动
    else {
        if (ty < 0){
            text = "向上滑动"
            this.data.currentGesture = 3

        }
```

```
      else if (ty > 0) {
        text = "向下滑动"
        this.data.currentGesture = 4
      }

    }

    //将当前坐标进行保存以进行下一次计算
    this.data.lastX = currentX
    this.data.lastY = currentY
    this.setData({
      text : text,
    });
  },

handletouchtart:function(event) {
```

```
    console.log(event)
    this.data.lastX = event.touches[0].pageX
    this.data.lastY = event.touches[0].pageY
  },
  handletouchend:function(event) {
    console.log(event)
    this.data.currentGesture = 0
    this.setData({
      text : "没有滑动",
    });
  },
})
```

多点触控

由于多点触控需要真机来测试，而我的 appid 还在申请中，只能延后讲解了。这里大概提下思路，比如**手指张开**的操作，可以分别判断两个触摸点的移动方向，比如靠左的触摸点向左滑动，靠右的触摸点向右滑动，即可判定为**手指张开**操作。