PTRScrollList 跨平台刷新组件

作者：HuixiaZhang
原文链接：https://ld246.com/article/1540546094585
来源网站：链滴

<img src="https://ld246.com/images/img-loading.svg" alt="" data-src="https://b3logfile
com/bing/20180505.jpg?imageView2/1/w/960/h/520/interlace/1/q/100">
<h2 id="PTRScrollList跨平台刷新组件">PTRScrollList 跨平台刷新组件</h2>
<hr>

<li class="vditor-task"><input disabled type="checkbox"> 平台支持

<li class="vditor-task vditor-task--done"><input checked disabled type="checkbox"> iOS</l
>
<li class="vditor-task vditor-task--done"><input checked disabled type="checkbox"> Andro
d

<li class="vditor-task"><input disabled type="checkbox"> 功能特性

<li class="vditor-task vditor-task--done"><input checked disabled type="checkbox"> 支持
拉刷新、加载更多
<li class="vditor-task vditor-task--done"><input checked disabled type="checkbox"> 支持
新组件自定义
<li class="vditor-task vditor-task--done"><input checked disabled type="checkbox"> 支持
新组件的动态显示与隐藏
<li class="vditor-task vditor-task--done"><input checked disabled type="checkbox"> 支持 c
nentsInset 属性

<li class="vditor-task"><input disabled type="checkbox"> 组件扩展

<li class="vditor-task vditor-task--done"><input checked disabled type="checkbox"> ScrollLi
t
<li class="vditor-task vditor-task--done"><input checked disabled type="checkbox"> ListVi
w
<li class="vditor-task vditor-task--done"><input checked disabled type="checkbox"> FlatLis

<li class="vditor-task vditor-task--done"><input checked disabled type="checkbox"> Virtual
zedList

注：Android 的支持正在实现中
<img src="https://raw.githubusercontent.com/bird-xiong/PTRScrollList/master/res/11.gif
 alt="刷新动效">
<h4 id="JSX-属性">JSX 属性</h4>
<pre><code class="highlight-chroma"><span class="highlight
cl"><PTRScrollList
  data={this.
tate.data}
  onHeader
efreshing={this._onHeaderRefreshing}
  onFooterR
freshing={this._onFooterRefreshing}
  scrollCom
onent={"FlatList"}
  enableFoo
erInfinite={this.state.enableFooterInfinite}

```
</span></span><span class="highlight-line"><span class="highlight-cl">          enableHea
erRefresh={this.state.enableHeaderRefresh}
</span></span><span class="highlight-line"><span class="highlight-cl">          keyExtract
r={this._keyExtractor}
</span></span><span class="highlight-line"><span class="highlight-cl">          renderIte
={({ item, index })=&gt;this._renderRower(item, index)}
</span></span><span class="highlight-line"><span class="highlight-cl">          ref={ref =
gt;this.ptrScrollList = ref}/&gt;
</span></span></code></pre>
<blockquote>
<p><code>onHeaderRefreshing</code></p>
<h5 id="头部开始刷新的回调方法"><em>头部开始刷新的回调方法</em></h5>
</blockquote>
<blockquote>
<p><code>onFooterRefreshing</code></p>
<h5 id="底部刷新的回调方法"><em>底部刷新的回调方法</em></h5>
</blockquote>
<blockquote>
<p><code>scrollComponent</code></p>
<h5 id="指定扩展的组件类型"><em>指定扩展的组件类型</em></h5>
</blockquote>
<blockquote>
<p><code>enableFooterInfinite</code></p>
<h5 id="隐藏底部"><em>隐藏底部</em></h5>
</blockquote>
<blockquote>
<p><code>enableHeaderRefresh</code></p>
<h5 id="隐藏头部"><em>隐藏头部</em></h5>
</blockquote>
<blockquote>
<p><code>renderFooterInfinite</code></p>
<h5 id="自定义底部刷新"><em>自定义底部刷新</em></h5>
</blockquote>
<blockquote>
<p><code>renderHeaderRefresh</code></p>
<h5 id="自定义头部刷新"><em>自定义头部刷新</em></h5>
</blockquote>
<h4 id="API">API</h4>
<p>1. 头部刷新成功</p>
<blockquote>
<p><code>this.ptrScrollList.ptr_headerRefreshFinished()</code></p>
</blockquote>
<h5 id="参数可缺省-传入false时可重置当前列表刷新状态"><em>参数可缺省，传入 false 时可重
当前列表刷新状态</em></h5>
<p>2. 底部刷新成功</p>
<blockquote>
<p><code>this.ptrScrollList.ptr_footerRefershFinished()</code></p>
</blockquote>
<h5 id="参数可缺省-传入false时可隐藏底部刷新组件"><em>参数可缺省，传入 false 时可隐藏底
刷新组件</em></h5>
<h4 id="自定义头部刷新-">自定义头部刷新</h4>
<p>目前项目中自带一个刷新组件 <code>HeaderComponent</code>，但是通常来说我们总是
定制化的需求，当前的刷新样式是使用 lottie 库生成的一个 json 文件，你可以去<a href="https://ld
46.com/forward?goto=https%3A%2F%2Fwww.lottiefiles.com%2Ftag%2Floading" target="_bl
```

nk" rel="nofollow ugc">这里找找有没有你喜欢的刷新样式
目前你有两种方式替换它
1. 重写 <code>HeaderComponent</code> 里面的样式
2. 使用 <code>renderHeaderRefresh</code> 属性覆盖它
这个时候，在你的自定义组件中需要实现一下几个方法
<blockquote>
<code>hc_refreshFinished = () => {...};</code>
<h5 id="在刷新完成时被调用">在刷新完成时被调用</h5>
</blockquote>
<blockquote>
<code>hc_startLoading = () => {...};</code>
<h5 id="在开始刷新的时候被调用">在开始刷新的时候被调用</h5>
</blockquote>
<blockquote>
<code>hc_updateProgress = per => {...};</code>
<h5 id="在滑动的过程中被调用-per-为当前滑动距离的百分比">在滑动的过程中被调用，per 为当前滑动距离的百分比</h5>
</blockquote>
<blockquote>
<code>hc_updateStatus = status => {...};</code>
<h5 id="在列表状态变化的时候被调用-status为刷新状态">在列表状态变化的时候被调用，status 为刷新状态</h5>
</blockquote>
<blockquote>
<code>hc_resetStatus = () => {...};</code>
<h5 id="刷新失败或者重置整个组件时被调用-这时候应该停止当前所有的动画并还原响应的状态">刷新失败或者重置整个组件时被调用，这时候应该停止当前所有的动画并还原响应的状态</h5>
</blockquote>
当你的动画执行完成后，需要告诉 PTRScrollList
<code>this.props.ptrScrollFinished && this.props.ptrScrollFinished();</code>
<h4 id="自定义底部刷新-">自定义底部刷新</h4>
底部组件跟头部组件一样，也有个内置的默认组件样式，你可以修改它或者通过 <code>renderHeaderRefresh</code> 覆盖它，通常情况下底部样式比较简单，只需要一点点文字
<pre><code class="highlight-chroma">class FooterComponent extends Component {
  render() {
    return <Text style={[Styles.loadMoreFont]}>{"正在加载..."}</Text>;
  }
}
</code></pre>
<h4 id="Android-的手势事件冲突">Android 的手势事件冲突</h4>
<h5 id="下拉事件的处理在Android上面实现起来比iOS上困难的多-不仅仅是因为Android不支持scrollview的弹性效果-而是在用手势处理时-你会发现目前RN的线程模型无法处理原生控件的手势事件-scollview-和自定义的手势事件之间的冲突-这个是一个框架上的瓶颈问题-据说在RN的重构版中会解决类问题-">下拉事件的处理在 Android 上面实现起来比 iOS 上困难的多，不仅仅是因为 Android 不持 scrollview 的弹性效果，而是在用手势处理时，你会发现目前 RN 的线程模型无法处理原生控件的势事件（scrollview）和自定义的手势事件之间的冲突，这个是一个框架上的瓶颈问题，据说在 RN 重构版中会解决这类问题。
</h5>
<h5 id="这个其实很好理解-事件在传递链中必须选择截获当前事件或者传递当前事件-当scrollview手势事件和自定义的手势事件同时存在时-我们也希望能按这种标准处理-但是很显让因为js线程是异的-scrollview的手势不会等待底层视图的事件拦截器的处理结果-在RN的生态体系中-你会发现scrollvi

w的事件总是优先级高的-并且你无法改变它-">这个其实很好理解，事件在传递链中必须选择截获当事件或者传递当前事件，当 scrollview 的手势事件和自定义的手势事件同时存在时，我们也希望能按种标准处理，但是很显让因为 js 线程是异步的，scrollview 的手势不会等待底层视图的事件拦截器的理结果，在 RN 的生态体系中，你会发现 scrollview 的事件总是优先级高的，并且你无法改变它。<b></h5>
<img src="https://raw.githubusercontent.com/bird-xiong/PTRScrollList/master/res/zenul.png" alt="传递链">
<h5 id="所以在PTRScrollList中做一个比较取巧的方法-通过禁用scrollview的滚动来使得我们可以获整个手势事件流-才能控制头部组件的下拉状态-">所以在 PTRScrollList 中做一个比较取巧的方法通过禁用 scrollview 的滚动来使得我们可以截获整个手势事件流，才能控制头部组件的下拉状态。<b></h5>
github 查看源码