0、手把手教React Native实战之开山篇

作者简介

东方耀 Android开发 RN技术 facebook github
android ios 原生开发 react reactjs nodejs 前端 进入 移动互联网 js nodejs 大波 app 个人角度 学习的必要性 全栈工程师的捷径

公司角度 组件化 成本降低 热更新

加作者微信公众号（dongfangyao888）或扫描下面二维码

推送高清视频教程+语音解说+课堂笔记和源码

技术背景

app store facebook html5 native app Hybrid app native + web 混合模式

视频课程简介

1.基础语法

2.API和组件

3.App更新 热更新上架

4.实战项目 3个 RN技术开发

0、配套视频(下载地址)：https://yunpan.cn/cY4JWzTtmVyNY 访问密码 7b60 或 http://vdisk.weibo.com/s/aLDC43gEH4wZV

12、手把手教React Native实战之View组件

在React Native里有一个类似于div的组件，那就是View组件，支持多层嵌套，支持flexbox布局

实例步骤：

1.加载View组件

2.创建组件

3.添加样式表

4.注册入口

5.外层布局

```
    <View style={styles.item}></View>
    <View style={styles.item}></View>
    <View style={styles.item}></View>

  </View>
```

6.flexbox水平三栏布局

外联样式：style={styles.container}

内联样式：style={{flex:1,borderWidth:1,borderColor:'red',flexDirection:'row'}}

多个样式：style= {[styles.container,styles.flex,{borderWidth:1,borderColor:'red'}]}

7.上下两栏布局

团购

```
      <View style={[styles.center,styles.flex]}>
        <Text>客栈，公寓</Text>

      </View>
```

8.完善效果

```
<View style={[styles.item,styles.lineLeftRight]}>

      <View style={[styles.center,styles.flex,styles.lineCenter]}>
        <Text style={styles.font}>海外酒店</Text>
      </View>

      <View style={[styles.center,styles.flex]}>
        <Text style={styles.font}>特惠酒店</Text>

      </View>



    </View>
```

12、配套视频(下载地址)：https://yunpan.cn/cqYXSpE5UsAZ4 访问密码 2af4 或 http://vdisk.weibo.com/s/aLDC43gEHncVH

13、手把手教React Native实战之Text组件

Text组件主要用于显示文本；具有响应性，可以嵌套，可以继承样式

内部Text组件可以继承外部Text组件的样式

Text组件的特性：

1.onPress

2.numberOfLines 最多显示多少行

3.onLayout

案例：网易新闻客户端 Text组件实现

组件的颗粒度设计主要取决于应用的结构设计

1.头部组件 单独封装 独立成一个文件

```
module.exports=Header;
```

```
const Header=require('./header');
```

2.列表组件

```
<List title='一线城市楼市退烧 有房源一夜跌价160万'></List>
```

3.重要新闻 组件

```
<ImportantNews news={[ '解放军报报社大楼正在拆除 标识已被卸下(图)', '韩国停签东三省52家旅行社 或为阻止朝旅游创汇', '南京大学生发起亲吻陌生人活动 有女生献初吻-南京大学生发起亲吻陌生人活动 有女生献初吻-南京大学生发起亲吻陌生人活动 有女生献初吻', '防总部署长江防汛:以防御98年量级大洪水为目标' ]}> </ImportantNews>
```

`<Text onPress={this.show.bind(this,this.props.news[i])} numberOfLines={2} style={styles.news_item}

```
                >{this.props.news[i]}</Text>`
```

警告的处理：数组里面需要key属性 参考：https://fb.me/react-warning-keys 需要翻墙（免费的翻墙浏览器 QQ群共享）

13、配套视频(下载地址)：https://yunpan.cn/cqP9zxBF5tEjU 访问密码 4838 或者 http://vdisk.weibo.com/s/aLDC43gEHmN6c

13、配套视频(补充)：https://yunpan.cn/cqPUEU4RjiKTl 访问密码 5814 或者 http://vdisk.weibo.com/s/aLDC43gEHmMRC

14、手把手教React Native实战之Navigator组件初探

一个应用往往是由多功能视图组成！多页面的切换：路由或导航

在RN中专门负责视图切换的组件：Navigator,NavigatorIOS

导航器对比：

Navigator和NavigatorIOS都可以用来管理应用中“场景”的导航（也可以称作屏幕）。导航器建立了一个路由栈，用来弹出，推入或者替换路由状态。它们和html5中的history API很类似。主要的区别在于NavigatorIOS使用了iOS中的UINavigationController类，而Navigator则完全用js重写了一个类似功能的React组件。因此Navigator可以兼容iOS和Android，而NavigatorIOS只能用于iOS。

NavigatorIOS 轻量、受限的API设置，使其相对Navigator来说不太方便定制。 由开源社区主导开发 —— React Native的官方团队并不在自己的应用中使用。 对于大多数正式的App开发，我们建议使用 Navigator —— 使用NavigatorIOS实现复杂的需求很容易碰到麻烦。

导航器通过路由对象来分辨不同的场景。利用renderScene方法，导航栏可以根据指定的路由来渲染场景。 可以通过configureScene属性获取指定路由对象的配置信息，从而改变场景的动画或者手势。

最后的几行: renderScene={(route, navigator) => { let Component = route.component; return <Component {...route.params} navigator={navigator} /> }} />);

这里是每个人最疑惑的，我们先看到回调里的两个参数:route, navigator。通过打印我们发现route里其实就是我们传递的name,component这两个货，navigator是一个Navigator的对象，为什么呢，因为它有push pop jump...等方法，这是我们等下用来跳转页面用的那个navigator对象。

return <Component {...route.params} navigator={navigator} /> 这里有一个判断，也就是如果传递进来的component存在，那我们就是返回一个这个component，结合前面 initialRoute 的参数，我们就是知道，这是一个会被render出来给用户看到的component，然后navigator作为props传递给了这个component。

所以下一步，我们可以直接拿到这个 props.navigator:

14、配套视频(下载地址)：https://yunpan.cn/cqGDYIBDYr5rB 访问密码 cdec

15、手把手教React Native实战之Navigator参数传递

怎么传递参数过去，或者从对方获取参数。

从列表页To详情页：

传递参数，通过push就可以了。

params: { id: this.state.id }

这个语法是把 routes.params 里的每个key 作为props的一个属性：{...route.params}

componentDidMount() { //这里获取从列表页传递过来的参数: id this.setState({ id: this.props.id }); }

使用这个参数：{this.state.id}

从详情页To列表页：

综合实例：从列表页传递用户id给详情页，详情页跟进用户id查询用户信息，并把 用户信息回传给列表页显示出来

const self=this;

params:{ id:this.state.id, //从详情页获取user getUser: function(user) { self.setState({ user: user }) } }

if(this.props.getUser) { let user = USER_MODELS[this.props.id]; this.props.getUser(user); }

```
    if(navigator) {
        //很熟悉吧，入栈出栈~ 把当前的页面pop掉，这里就返回到了上一个页面:List了
        navigator.pop();
    }
```

15、配套视频(下载地址)：https://yunpan.cn/cqGhLTHhhZnP2 访问密码 a8bd

16、手把手教React Native实战之TextInput组件

文本输入框：基本组件 自动补全的搜索功能

调试apk目录：DongFang\android\app\build\outputs\apk

TextInput的主要属性和事件如下：

autoCapitalize：枚举类型，可选值有none sentences words characters 当用户输入时，用于提示

placeholder：占位符，在输入前显示的文本内容

value：文本输入框的默认值

placeholderTextColor：占位符文本的颜色

password：boolean类型 true表示密码输入显示*

multiline：多行输入

editable：文本框是否可输入

autoFocus:自动聚焦

clearButtonMode：枚举类型，never while-editing unless-editing always 用于显示清除按钮

maxLength：能够输入的最长字符数

enablesReturnKeyAutomatically：如果为true 表示没有文本时键盘是不能有返回键的，其默认值为false

returnKeyType：枚举类型 default go google join next route search send yahoo done emergency-call 表示软键盘返回键显示的字符串

secureTextEntry：隐藏输入内容，默认值为false

onChangeText：当文本输入框的内容变化时，调用改函数；onChangeText接收一个文本的参数对象

onChange：当文本变化时，调用该函数

onEndEditing：当结束编辑时，调用改函数

onBlur：失去焦点触发事件

onFocus：获得焦点时触发事件

onSubmitEditing：当结束编辑后，点击键盘的提交按钮触发该事件

案例：搜索框

16、配套视频(下载地址)：https://yunpan.cn/cSFjnXwyzXRip 访问密码 ef33

17、手把手教React Native实战之TextInput自动提示的搜索框

接着16讲的搜索框继续，加上自动提示的功能

很多app的搜索都是这样的：我们输入一个关键词的时候，会列出相关的搜索结果列表，一般情况下，完成该功能需要一个搜索服务，返回N条结果，并将其展示出来;这里我们用静态数据模拟出来。

17、配套视频(下载地址)：https://yunpan.cn/cS3vKXfQunD39 访问密码 52b9

18、手把手教React Native实战之Touchable类组件

React Native没有像Web开发那样给元素绑定click事件，前面讲的Text组件有onPress事件回调，View组件是没有的，但是在实际项目开发中，很多其他的组件也是需要可以被点击的，RN提供了3个组件来赋予被点击的能力（去包裹其他组件即可），这3个组件被称为“Touchable类组件”：

1.TouchableHighlight 高亮

属性：activeOpacity(设置透明度)、onHideUnderlay、onShowUnderlay、underlayColor(点击时背景阴影效果的背景颜色)

2.TouchableOpacity 透明度

属性：activeOpacity

3.TouchableWithoutFeedback 无反馈 不会出现任何视觉变化

不建议使用；属性：onLongPress、onPressIn、onPressOut

在app中我们希望点击的时候会有一些视觉上的变化，这个变化会提醒我们已经点击过了，从而避免重复点击

18、配套视频(下载地址)：https://yunpan.cn/cS3vW9nauRNxa 访问密码 6a08

19、手把手教React Native实战之图片Image组件

React Native的Image组件调用的图片途径比较多：网络图片、本地图片、照相机图片

Image组件的属性：

resizeMode：图片适用的模式 cover、contain、stretch

source:图片的引用地址

网络图片：source={{uri:'http://.png'}}

本地图片：source={require('./baidulogo.png')}

静态图片资源：注意：如果你添加图片的时候packager正在运行，则你需要重启packager以便能正确引入新添加的图片。

这样会带来如下的一些好处:

iOS和Android一致的文件系统。 图片和JS代码处在相同的文件夹，这样组件就可以包含自己所用的图片而不用单独去设置。 不需要全局命名。你不用再担心图片名字的冲突问题了。 只有实际被用到（即被require）的图片才会被打包到你的app。 现在在开发期间，增加和修改图片不需要重新编译了，只要和修改js代码一样刷新你的模拟器就可以了。 与访问网络图片相比，Packager可以得知图片大小了，不需要在代码里再声明一遍尺寸。 现在通过npm来分发组件或库可以包含图片了。

注意：为了使新的图片资源机制正常工作，require中的图片名字必须是一个静态字符串。

使用混合App的图片资源

如果你在编写一个混合App（一部分UI使用React Native，而另一部分使用平台原生代码），也可以使用已经打包到App中的图片资源（通过Xcode的asset类目或者Android的drawable文件夹打包）

<Image source={{uri: 'app_icon'}} style={{width: 40, height: 40}} />

注意：这一做法并没有任何安全检查。你需要自己确保图片在应用中确实存在，而且还需要指定尺寸

注意：网络图片，你需要手动指定图片的尺寸

关于图片的尺寸，React Native会自动为你选好。如果没有，则会选择最接近的尺寸进行缩放，但也至少缩放到比所需尺寸大出50%，以使图片看起来仍然足够清晰。这一切过程都是自动完成的，所以你不用操心自己去完成这些繁琐且易错的代码。

19、配套视频(下载地址)：https://yunpan.cn/cS3ekcwmwhLSz 访问密码 2023

20、手把手教React Native实战之Picker组件和箭头函数

本组件可以在iOS和Android上渲染原生的选择器（Picker）

20、配套视频(下载地址)：https://yunpan.cn/cSueHJKakErf2 访问密码 f349

21、手把手教React Native实战之ProgressBar组件

分平台的：ProgressBarAndroid ProgressViewIOS

styleAttr：进度条的样式 Horizontal Small Large Inverse SmallInverse LargeInverse

progress：当前的进度值（在0到1之间）

21、配套视频(下载地址)：https://yunpan.cn/cSx7C5nTQWx5q 访问密码 648b

22、手把手教React Native实战之DrawerLayoutAndroid组件

封装了平台DrawerLayout（仅限安卓平台）的React组件。抽屉（通常用于导航切换）是通过renderNavigationView方法渲染的，并且DrawerLayoutAndroid的直接子视图会成为主视图（用于放置你的内容）。导航视图一开始在屏幕上并不可见，不过可以从drawerPosition指定的窗口侧面拖拽出来，并且抽屉的宽度可以使用drawerWidth属性来指定。

drawerPosition：DrawerLayoutAndroid.positions.Right 左右 默认左

drawerWidth：指定抽屉的宽度，也就是从屏幕边缘拖进的视图的宽度

onDrawerClose

onDrawerOpen

onDrawerSlide：每当导航视图（抽屉）产生交互的时候调用此回调函数

onDrawerStateChanged：idle（空闲） dragging（拖拽中） settling（停靠中）

renderNavigationView：渲染一个可以从屏幕一边拖入的导航视图

drawerLockMode：设置抽屉的锁定模式 有三种模式：

1.unlocked (默认值)，意味着此时抽屉可以响应打开和关闭的手势操作

2.locked-closed，意味着此时抽屉将保持关闭，不可用手势打开。

3.locked-open，意味着此时抽屉将保持打开，不可用手势关闭。

无论抽屉处于那种状态，都仍然可以调用openDrawer/closeDrawer这两个方法打开和关闭

22、配套视频(下载地址)：https://yunpan.cn/cSx7azQYEez7g 访问密码 de38

23、手把手教React Native实战之ViewPagerAndroid组件初探

一个允许在子视图之间左右翻页的容器。每一个ViewPagerAndroid的子容器会被视作一个单独的页，并且会被拉伸填满ViewPagerAndroid。

注意所有的子视图都必须是纯View，而不能是自定义的复合容器

属性：

1.initialPage 初始选中页的下标

2.onPageScroll 当在页间切换时（不论是由于动画还是由于用户在页间滑动/拖拽）执行

回调参数中的event.nativeEvent对象会包含如下数据： position offset 一个在[0,1)（大于等于0，小于1）之间的范围，代表当前页面切换的状态

3.onPageScrollStateChanged idle dragging settling

4.onPageSelected 这个回调会在页面切换完成后（当用户在页面间滑动）调用

回调参数中的event.nativeEvent对象会包含如下的字段： position

5.scrollEnabled boolean

23、配套视频(下载地址)：https://yunpan.cn/cS6VtFy7F3CVE 访问密码 68d1

24、手把手教React Native实战之ViewPagerAndroid做引导页

开始做个小项目(DfyProject01) 来将知识点串起来

ViewPagerAndroid做引导页

我们能学到：

1.组件如何联动

2.符合ES6的标准写法

3.组件的封装 非常灵活

4.Navigator路由的用法

24、配套视频(下载地址)：https://yunpan.cn/cSkk5qHjkRLEz 访问密码 f305

25、手把手教React Native实战之仿异步获取网络数据

进入首页HomeUI获取网络数据

Android原生开发：子线程 handler机制 异步任务AsyncTask

ReactNative 擅长UI界面处理 通过this.state来触发

RN中的网络请求：XMLHttpRequest Fetch post get

MVC mListView.setAdapter(new myAdapter)

Item封装成一个组件

25、配套视频(下载地址)：https://yunpan.cn/cS2TlyZEaKs9m 访问密码 f8f4

26、手把手教React Native实战之开源轮播组件react-native-swiper

github上搜索 react-native-swiper

在项目的根目录（即package.json文件所在的目录）

npm的命令：

安装模块：npm i react-native-swiper --save

查看模块：npm view react-native-swiper

删除模块：npm rm react-native-swiper --save

查看帮助命令：npm help 命令

入口组件：DfyProject01

注意：设置轮播组件Swiper的包裹容器高度，使用属性设置，不能通过样式设置

实现滚动视图：ScrollView 计算步长

26、配套视频(下载地址)：https://yunpan.cn/cS5vMhdBG3Wat 访问密码 e6a9

27、手把手教React Native实战之WebView组件

Hybrid App（混合模式移动应用）是指介于web-app、native-app这两者之间的app，兼具“Native App良好用户交互体验的优势”和“Web App跨平台开发的优势”。

重点理解：为什么rn中会有WebView？

automaticallyAdjustContentInsets：是否自动调整内部内容

bounces（IOS）：回弹效果 如果为false，则内容拉到底部或头部不回弹，默认为true

domStorageEnabled(Android)：仅限Android平台。指定是否开启DOM本地存储

javaScriptEnabled：仅限Android平台。iOS平台JavaScript是默认开启的

contentInset：内部内容偏移值 该值为一个JavaScript对象 {top:number,left:number,bottom:number,right:number}

source:{{uri:'网址'}}在WebView中载入一段静态的html代码或是一个url（还可以附带一些header选项）{{html:'html代码块'}}

injectedJavaScript:注入的js代码，其值为字符串，如果加上了该属性，就会在webview里面执行js代码（在网页加载之前注入）

mediaPlaybackRequiresUserAction：设置页面中的HTML5音视频是否需要在用户点击后再开始播放。默认值为false

onNavigationStateChange：监听导航状态变化的函数（当发现浏览器地址改变时，触发事件）

renderError：监听渲染页面出错的函数

startInLoadingState：是否开启页面加载的状态

renderLoading:webview组件正在渲染页面时触发的函数，需要同startInLoadingState一起使用，当startInLoadingState为true时该函数才起作用

scrollEnabled（IOS）：表示webview里面页面是否能滚动，如果其值为true则可以滚动，否则禁止滚动

scalesPageToFit：按照页面比例和内容宽高比例自动缩放内容

27、配套视频(下载地址)：https://yunpan.cn/cSeEkqgveNvJ8 访问密码 3207

28、手把手教React Native实战之常用IDE开发工具

Web前端：WebStorm Sublime 都是收费的

FaceBook官方：nuclide 只支持Mac 基于Atom（github的）（Atom最大的特色就是可以安装很多的插件来完成我们的需求）炫酷插件

跨平台免费的：微软老大哥的 Visual Studio Code 免费的

Visual Studio Code (简称 VS Code / VSC) 是一款免费开源的现代化轻量级代码编辑器，支持几乎所有主流的开发语言的语法高亮、智能代码补全、自定义热键、括号匹配、代码片段、代码对比 Diff、GIT 等特性，支持插件扩展，并针对网页开发和云端应用开发做了优化。软件跨平台支持 Win、Mac 以及 Linux，运行流畅，可谓是微软的良心之作

Visual Studio Code 现在已经支持大量的扩展插件，大家可以根据自己的需求打造出最适合自己使用的编辑器了 插件：https://marketplace.visualstudio.com/VSCode

28、配套视频(下载地址)：https://yunpan.cn/cSUnwQGHxKrm6 访问密码 75ee

29、手把手教React Native实战之ListView组件

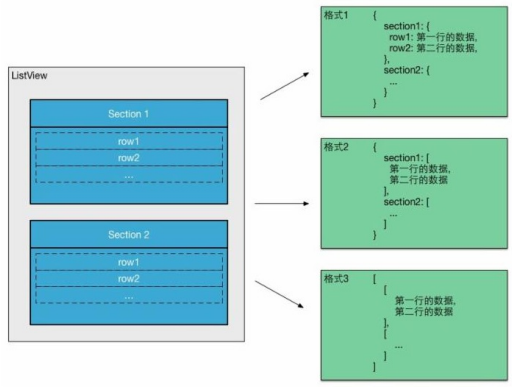
原生：mListView mAdapter List集合数据 MVC

ListView组件是React Native中一个比较核心的组件，用途非常的广。该组件设计用来高效的展示垂直滚动的数据列表。最简单的API就是创建一个ListView.DataSource对象，同时给该对象传入一个简单的数据集合。并且使用数据源(data source)实例化一个ListView组件,定义一个renderRow回调方法(该方法的参数是一个数组)，该renderRow方法会返回一个可渲染的组件(该就是列表的每一行的item)

由于平台差异性，React Native 中的滚动列表组件 ListView 并没有直接映射为 android 中的 ListView 或 iOS 中的 UITableView，而是在ScrollView 的基础上使用 JS 做了一次封装。这样，滚动体验部分由 Native 负责，而 React 部分则专注于组件何时渲染、如何渲染等问题。

数据源默认的格式有三个维度：

第一个维度是 sectionId ，标识属于哪一段， 可以手动指定或隐式地使用数组索引或对象的 key 值；第二个维度是 rowId ，标识某个数据段下的某一个行，同样可以手动指定或隐式地使用数组索引或对象的 key 值； 第三个维度是具体的数据对象，根据实际的需要而定。 需要注意的是，上面只是 默认的数据格式 ，如果它不符合实际的需求， 完全可以使用自定义的数据结构 。唯一的区别就是需要额外指定给 ListView 数据源中哪些是 id，哪些是 rowData。



DataSource 的构造函数接收以下4个参数（都是可选）： rowHasChanged : 用于在数据变化的时候，计算出变化的部分，在更新时只渲染脏数据； sectionHeaderHasChanged : 同理，在列表带分段标题时需要实现； getRowData/getSectionHeaderData : 如果遵循默认的数据源格式，这两个方法就没有必要实现，用内部默认的即可；而当数据源格式是自定义时，需要手动实现这两个方法。

DataSource 的初始化一般在 constructor 方法中

数据源确定后，下一个工作就是列表的渲染。在渲染时发挥重要作用的是 renderRow 属性，它接收数据源中保存的数据对象，并通过返回值确定该行该如何进行展现。我们可以对所有行统一进行展现，也可以根据里面的字段做出不同的展现。在列表包含 sectionHeader 时，还需要实现 renderSectionHeader 方法。

注意

要更新datasource中的数据，请（每次都重新）调用cloneWithRows方法（如果用到了section，则对应cloneWithRowsAndSections方法）。数据源中的数据本身是不可修改的，所以请勿直接尝试修改。clone方法会自动提取新数据并进行逐行对比（使用rowHasChanged方法中的策略），这样ListView就知道哪些行需要重新渲染了。

React Native ListView sticky效果实现

dataBlob{The dataBlob is a data structure (generally an object) that holds all the data powering the ListView}：dataBlob 包含ListView所需的所有数据（section header 和 rows），在ListView渲染数据时，使用 getSectionData 和 getRowData 来渲染每一行数据。 dataBlob 的 key 值包含 sectionID + rowId

Data Blob	
ID1	“Section 1”
ID2	“Section 2”
…	…
ID{N}	“Section {N}”
“ID1:rowID0”	{Row Data for Section 1 -> Row 1}
“ID1:rowID1”	{Row Data for Section 1 -> Row 2}
“ID2:rowID0”	{Row Data for Section 2 -> Row 1}
…	{…}
“ID{N}:…”	{Row Data for Last Section & Row}

sectionIDs[]：用于标识每组section

Section IDs	
0	ID1
1	ID2
2	ID3
3	ID4
…	…
n	ID{N}

rowIDs[]：用于描述每个 section 里的每行数据的位置及是否需要渲染。在ListView渲染时，会先遍历 rowIDs 获取到对应的 dataBlob 数据。

Row IDs	
0	[rowID0, rowID1, rowID2, …, rowID{N}]
1	[…]
2	[…]
3	[…]
…	…
n	[…]

片段头sticky失效原因：

1.Found this out by accident, but if you put your ListView inside a ScrollView, the headers will not be sticky

2.iOS RCTScrollView implements sticky headers 而android没有

3.ListView的Android版本不支持sticky-header特性

4.用这个替代：https://github.com/emilsjolander/StickyScrollViewItems

29、配套视频(下载地址)：https://yunpan.cn/cSIci374GY5IE 访问密码 0af6

30、手把手教React Native实战之ListView高级特性

分页

当数据量很大的时候如何分页加载 。这种情形分两种情况考虑：

1.数据一次性拿到，边滚动边加载

2.数据不是一次性拿到，而是有可能分屏取数据

对于第一种情况，在 ListView 内部其实已经做了分页的处理：

ListView 内部通过 curRenderedRowsCount 状态保存已渲染的行数； 初始状态下，要加载的数据条数等于 initialListSize （默认为 10 条）； 在滚动时检测当前滚动的位置和最底部的距离，如果小于 scrollRenderAheadDistance (默认为 1000)，就更新 curRenderedRowsCount ，在它原有值基础上加 pageSize 个（默认为 1 条）； 由于属性变化，触发了 ListView 重新的 render 。在渲染过程中，curRenderedRowsCount 起到截断数据的作用，React 的 diff 算法使得只有新加入的数据才会渲染到了界面上。 整个过程类似于 Web 端懒加载机制，即 每次在和底部的距离达到一个阈值时，加载接下来的 pageSize 个数据 。

对于第二种情况，ListView 提供了相关的属性：

onEndReachedThreshold ，在滚动即将到达底部时触发； onEndReached ，在已经到达底部时触发；

我们可以在这两个方法中调用接口去拿数据，取到数据后再更新数据源。

多列(Grid效果)

很多页面中的列表并非单列的，乍一看似乎要做出不少调整，但实际上只通过布局即可达到相关效果。ListView 并没有强制要求一个 rowData 在展示时一定要占满一行，在多列的情况下，我们适时调整每个 rowData 占据的宽度即可。

由于 React Native 使用 Flexbox 进行布局，给ListView设置属性contentContainerStyle：在实现多列时，主要用到的是 flexWrap：wrap 属性：它的效果类似于 float，即水平地排列每一项，当放不下时进行折行处理。在设置每行视图占据一半宽度后就达到了两列的效果，多列的类似。

滚动

ListView 只是整合了数据和展现，但实际滚动的功能还是由 ScrollView 全权负责。ScrollView 实现完全和平台相关：在 iOS 上，它映射为 RCTScrollView ；在 Android 上，它映射为 RCTScrollView 和 AndroidHorizontalScrollView 。

React Native 让不同端上的技术融合在了一起，同时也给开发人员提出了更高的要求。以 ScrollView 为例，大量的属性其实原封不动映射给了 UIScrollView ，这就意味着如果想再深入地研究下去，必须对客户端相关技术有足够了解。无论是前端还是客户端，跳出自己熟悉的那片领域也许才是更进一步的关键。

谈到滚动，有一点不得不说的就是 列表的无限加载 ，这牵涉到滚动的性能。

Github 上的这个 issue 对此展开了热烈的讨论。其中有人就提到，数据量很大情况下，ListView 在加载时所占用的 CPU 和内存会大大增加，滚动到最后就导致了应用 crash。

为此，ListView 中新添加了一个实验性的属性： removeClippedSubviews ，它能在滚动时及时删掉列表中处于视窗的之外的行，以此达到降低内存消耗的目的。不幸的是，即使设置了这个属性，程序虽然各项占用减少了不少，但还是没避免崩溃的命运。处于好奇，我也在最新版的 ListView 基础上做了简单尝试，不断加载一个无限大的列表，但并没有出现崩溃的情况：

即使加载了 3000、4000 行，Android 真机、iOS 真机和 iOS 模拟器上都没有崩溃； Android 上明显感到数据加载有 阶段性的延时 ，即滚动一定程度后，再次滚动数据始终加载不出来或要等一段时间才加载出来，体验较差；iOS 相比要流畅的多； 但不崩溃并非最终的目的，很多 React Native 使用者都在试图改进 ListView 的性能表现，相比于直接使用 Native 端的组件，ListView 性能还是差强人意，有很大优化空间。

总结

ListView 并没有创造出新的东西，它只是集各家所长，很好地将 React 的视图渲染和 Native 端很成熟的滚动机制融合在了一起，使用起来和其他组件无差，静态地定义View、动态地组织数据，是给人带来的直观感受。

30、配套视频(下载地址)：https://yunpan.cn/cSln34RprDCr3 访问密码 c927

未完待续。。。作者：东方耀 QQ：309623978