JDReact基础组件介绍

欢迎使用JDReact！在使用JDReact组件开发的过程中，我们也会使用一些RN的基础组件，下面先简单介绍一下。

RN基础组件介绍

> *View*

作为创建UI时最基础的组件，View是一个支持Flexbox布局、样式、一些触摸处理、和一些无障碍功能的容器，并且它可以放到其它的视图里，也可以有任意多个任意类型的子视图。不论在什么平台上，View都会直接对应一个平台的原生视图。（没接触RN的同学可以把View类比成html中的div，用来包裹其他组件）

APIDemos示例：

```
<View style={styles.wrapper} >
                <NavigationBar Title="View组件" showMoreNav={true} moreNavList={moreNavList} />
                <View style={styles.content}>
                    <JDText>
                        作为创建UI时最基础的组件，View是一个支持Flexbox布局、样式、一些触摸处理、和一些无障碍功能的
                    </JDText>


                    <JDText style={{ marginTop: 30 }}>flex-direction: row</JDText>
                    <JDText>横向布局</JDText>
                    <View style={styles.rowStyle}>
                        <View style={styles.greenBlock} />
                        <View style={styles.blueBlock} />
                        <View style={styles.grayBlock} />
                    </View>
                    <JDText>flex-direction: column</JDText>
                    <JDText>纵向布局(默认)</JDText>
                    <View style={styles.columnStyle}>
                        <View style={styles.greenBlock} />
                        <View style={styles.blueBlock} />
                        <View style={styles.grayBlock} />
                    </View>
                </View>

            </View>

            const styles = StyleSheet.create({
    wrapper: {
        flex: 1,

    },
    content: {
        paddingTop: 10,
        paddingLeft: 10,
        paddingRight: 10,
    },
    rowStyle: {
```

```
        flexDirection: 'row',
        justifyContent: 'center',
        alignItems: 'center',
        height: 80
    },
    columnStyle: {
        justifyContent: 'center',
        alignItems: 'center',
        height: 160
    },
    greenBlock: {
        width: 40,
        height: 40,
        backgroundColor: 'green',
    },
    blueBlock: {
        width: 40,
        height: 40,
        backgroundColor: 'blue',
    },
    grayBlock: {
        width: 40,
        height: 40,
        backgroundColor: 'gray',
    },

});
```

运行效果截图:

> *Text*

一个用于显示文本的React组件，并且它也支持嵌套、样式，以及触摸处理。（文字不可以直接写在View组件内，需要用Text组件包起来）

> *ScrollView*

一个包装了平台的ScrollView（滚动视图）的组件，同时还集成了触摸锁定的“响应者”系统。

ScrollView和ListView/FlatList应该如何选择？ScrollView会简单粗暴地把所有子元素一次性全部渲染出来。其原理浅显易懂，使用上自然也最简单。然而这样简单的渲染逻辑自然带来了性能上的不足。想象一下你有一个特别长的列表需要显示，可能有好几屏的高度。创建和渲染那些屏幕以外的JS组件和原生视图，显然对于渲染性能和内存占用都是一种极大的拖累和浪费。

性能：ScrollView < ListView < FlatList

>

FlatList

长列表推荐使用 `Flatlist` ,支持如下功能:

- 完全跨平台。
- 支持水平布局模式。
- 行组件显示或隐藏时可配置回调事件。
- 支持单独的头部组件。
- 支持单独的尾部组件。
- 支持自定义行间分隔线。
- 支持下拉刷新。
- 支持上拉加载。
- 支持跳转到指定行（ScrollToIndex）。

本组件实质是基于`<VirtualizedList>`组件的封装，因此也有下面这些需要注意的事项:

1. 当某行滑出渲染区域之外后，其内部状态将不会保留。请确保你在行组件以外的地方保留了数据。
2. 为了优化内存占用同时保持滑动的流畅，列表内容会在屏幕外异步绘制。这意味着如果用户滑动的速度超过渲染的速度，则会先看到空白的内容。这是为了优化不得不作出的妥协，而我们也在设法持续改进。
3. 本组件继承自PureComponent而非通常的Component，这意味着如果其props在浅比较中是相等的，则不会重新渲染。所以请先检查你的renderItem函数所依赖的props数据（包括data属性以及可能用到的父组件的state），如果是一个引用类型（Object或者数组都是引用类型），则需要先修改其引用地址（比如先复制到一个新的Object或者数组中），然后再修改其值，否则界面很可能不会刷新。（译注：这一段不了解的朋友建议先学习下js中的基本类型和引用类型。）
4. 默认情况下每行都需要提供一个不重复的key属性。你也可以提供一个keyExtractor函数来生成key。

“

Platform

Platform.OS在iOS上会返回'ios'，而在Android设备或模拟器上则会返回'android'，在web平台上回返回'web'

以上简单介绍了一下RN基础组件，具体api和其他组件可以参考中文文档

JDReact基础组件介绍

下面介绍一下RN中重要的基础组件，强烈推荐使用！！！！（基础组件的示例代码这里就不贴了，可以参照JDReactAPIDemos）

“

JDDevice

目前该组件封装了3个属性，4个方法，下面详细介绍:

```
JDDevice.width: 屏幕宽度

JDDevice.height:屏幕高度

JDDevice.dpr:屏幕dpr

getRpx(value):基于750屏幕宽度(iphone6)，按比例转换px

getDpx(value):基于dpr获取px

getFontSize(value):基于750屏幕宽度(iphone6)，计算字体大小

exitApp():退出应用
```

> *JDImage*

JDImage基于底层进行图片加载性能优化，修复了原生RN Image组件容易造成oom的问题

```
placeholder: 加载过程中或加载失败是显示的展位图片，可以是本地图片或远程图片（默认有一张带京东Logo的占位图）

hidden: 如果设置为true，将会用一个空白的view代替这张图片（业务里做内存回收可以使用）
```

> *JDRouter*

基于react-navagation封装的高性能跨平台路由跳转组件，路由配置写法类似react-router，非常简单易用，看APIDmoes的例子:

```
<Router navigationBar={(<NavigationBar />)}>

                <Route key="uis" component={UIIndex} type="resetTo" sceneConfig={Router.SceneConfigs.N
                <Route key="apis" component={ApiIndex} sceneConfig={Router.SceneConfigs.None} />
                            </Router>
```

- JDRouter 提供了 `push` , `pop`, `popTo` , `jumpTo`, `reset`, `resetTo`, `replace` 等等路由操作api
- JDRouter会为该组件增加新的生命周期 componentWillFocus 和 componentWillUnfocus,每次进入页面时都会调用componentWillFocus，而每次离开该页面时，会调用componentWillUnfocus, 如果componentWillUnfocus返回true,则停止离开当前页面。
- JDRouter提供统一的导航栏NavigationBar，该组件完美适配iphoneX,并且导航栏左侧默认提供回退按钮，右侧可以通过配置加入可配置的更多按钮
- 如果想使用导航方法，或者使用导航栏NavigationBar，必须要在项目的入口进行Router的配置，否则路由将不能初始化导致调用方法抛出异常。（有的业务只有一个页面，如果不初始化JDRouter的配置直接使用路由方法会抛出异常）

> *轮播图*

JDReact目前提供了5种样式的轮播图

1. JDEffectViewPager：支持三端的轮播图
2. JDExperimentalViewPager：支持ios和安卓
3. JDNativeCustomViewPager：支持ios和安卓
4. JDSwiper：支持三端
5. JDViewPager：支持三端

样式效果参考APIDemos

> *网络请求*

在JDReact中，网络请求调用客户端SDK发起的网络请求，需要引入的组件名为`JDNetwork`，目前提供三个api:

```
 /**
      * 通过京东客户端SDK进行网络请求
      * @param function_id: 请求GW的对应functionId，例如：getJpOrderDetail
      * @param host：正式环境的host地址，例如：android对应是'api.m.jd.com'，ios对应是'http://api.m.jd.com/clie
      * @param host_beta：预发布环境的host地址，例如：android对应是'beta-api.m.jd.com'，ios对应是'http://beta-
      * @param params_json：请求数据的参数，例如：{orderCode:123456}
      * @return Promise<resolve,reject>对象
      */
JDNetwork.fetch( function_id : string, host : string, host_beta : string, params_json : string, ): Pro

/**
      * 通过京东客户端SDK进行网络请求
      * @param function_id: 请求GW的对应functionId，例如：getJpOrderDetail
      * @param host：正式环境的host地址，例如：android对应是'api.m.jd.com'，ios对应是'http://api.m.jd.com/clie
      * @param host_beta：预发布环境的host地址，例如：android对应是'beta-api.m.jd.com'，ios对应是'http://beta-
      * @param params_json：请求数据的参数，例如：{orderCode:123456}
      * @param timeout：超时时间，例如：6000
      * @return Promise<resolve,reject>对象
      */
JDNetwork.fetchWithTimeOut( function_id : string, host : string, host_beta : string, params_json : str

/**
      * 通过京东客户端SDK进行网络请求，使用默认的Host地址
      * @param function_id: 请求GW的对应functionId，例如：getJpOrderDetail
      * @param params_json：请求数据的参数，例如：{orderCode:123456}
      * @return Promise<resolve,reject>对象
      */
JDNetwork.fetchWithoutHost( function_id : string, params_json : string, ): Promise
```

总结

更多组件介绍与使用请参考JDReact官方文档与JDReactAPIDemos。