一．环境搭建

1.安装jdk

2.安装sdk

3.安装C++环境

4.安装Node.js (https://nodejs.org/en/)

设置全局使用指定的镜像

npm config set registry https://registry.npm.taobao.org

npm config set disturl https://npm.taobao.org/dist

5.安装React Native命令行工具

github下载 (https://github.com/facebook/react-native)

npm install -g react-native-cli

6.创建项目

react-native init DongFang

7.运行packager 进入工程目录

react-native start

可以用浏览器访问http://localhost:8081/index.android.bundle?platform=android看看是否可以看到打包后的脚本

8.准备模拟器或真机 运行android

react-native run-android

1. React Native基础

1. react-native 中文网(http://reactnative.cn/) ,几乎所有的组件，api等都可以在上面找到。

2.目前来说使用ES6语法来替换ES5 语法

React Native 与Android原生映射表

|  |  |
| --- | --- |
| ReactNative | Android |
| Text | TextView |
| WebView | WebView |
| TextInput | EditText |
| Image | ImageView |
| ListView | ListView |
| ScrollView | ScrollView |
| ToastAndroid | Toast |
| AsyncStorage | SharePrefenence |

三，React Native 三方类库

1. react-native-image-picker 调用系统相册和图库返回数据  
2. react-native-tab-navigator 底部tab  
3.React-Native-ViewPager      https://github.com/zbtang/React-Native-ViewPager

4. react-native-viewpager      https://github.com/race604/react-native-viewpager

5. react-native-media-kit      网络视频播放

6. react-native-picker 选择器(时间,城市等)

7. react-native-picker-android 类似IOS时间选择器

8. react-native-datetime-picker 时间选择器

9. react-native-datetime 时间选择器

四,常用方法

//隐藏键盘

var TextInputState = TextInput.State;

TextInputState.blurTextInput(TextInputState.currentlyFocusedField());

//View控件

类似Android里的LinearLayout ，里面的子控件默认垂直排列，即

flexDirection： ‘column‘

如果要设置水平 ，设置

flexDirection： ‘row’

//点击事件

与Android 不同的事，react native 并没有一个类似button的控件，或者在控件上添加点击事件， react native有四个控件可以实现点击效果，但是只是包裹子空间，被包裹的子控件就具有点击功能，

TouchableHighlight 点击效果高亮

TouchableNativeFeedback 点击效果与原生效果一样

TouchableWithoutFeedback 无点击效果

TouchableOpacity 点击效果透明度改变

代码：

onPressButton(){

这里处理点击事件

}

<TouchableHighlight onPress={this.onPressButton}>

<Image

style={styles.button}

source={require('./button.png')}/>

</TouchableHighlight>

//调用一个控件的内部方法

<input type="text" ref={(ref) => this.myTextInput = ref} />

这里ref将this.myTextInput包装成input的对象，类似android里面的this.myTextInput = newinput();

下面是对象方法的调用

this.myTextInput.focus();

//控件属性的修改

React-Native 没有DOM的功能，即android里面直接获取控件进行属性的修改，而是通过this.state 来设置默认属性，通过setState()来修改属性，在render()函数中将需要的属性设置为this.state里面值，一旦调用setState()，会重新调用render()来渲染界面，从而达到属性值修改的效果

//React Native类的结构

//导系统的包

import React, {Component} from 'react';

import {

AppRegistry,

StyleSheet,

Navigator,

BackAndroid,

} from 'react-native';

//导入自定义的类

import Main from './app/Main';

var \_navigator;

//类的定义 跟java类似继承Component

class GossipTribe extends Component {

//构造函数，默认传个参数，需要先调用父类的方法

constructor(props) {

super(props);

}

//生命周期方法

componentDidMount() {

BackAndroid.addEventListener('hardwareBackPress', function () {

if (\_navigator && \_navigator.getCurrentRoutes().length > 1) {

\_navigator.pop();

return true;

}

return false;

});

}

//界面渲染，return返回需要显示的布局

render() {

定义默认加载的页面名称和组件

let defaultName='main';

let defaultComponent = Main;

return (

//页面跳转的导航器

<Navigator

style={styles.init}

定义路由，设置默认加载的页面

initialRoute={{name:defaultName,component:defaultComponent}}

定义页面跳转的东海效果

configureScene={() => Navigator.SceneConfigs.FadeAndroid}

渲染界面

renderScene={

(route,navigator)=>{

获得需要跳转的界面对象

let Component = route.component;

\_navigator = navigator;

遍历跳转界面传递的数据，并调到到新的界面

return <Component {...route.params} navigator={navigator}/>

}

}

/>

);

}

//ListView的使用

<ListView

style = { styles.container }

dataSource = { dataSource }

renderHeader = { this.renderHeader.bind(this) }

renderRow = { this.renderRow.bind(this) }

refreshControl={this.getRefreshControl(LATEST)}/>

dataSource: ListView的数据源，初始化的时候会有默认的数据源，

var dataSource = new ListView.DataSource({ rowHasChanged: (r1, r2) => r1 !== r2 });

如果需要新的数据源需要重新赋值

dataSource = dataSource.cloneWithRows(defaultTodos);

renderHeader ：ListView头部布局

renderRow ：ListView的Item布局和android类似

renderRow(dataRow) {}函数是Item布局，参数dataRow就是一个数据源的对象，类似android里面的一个对象，里面存放了对象的属性，用于在listview item上显示，或者触发点击事件。

refreshControl: ListView 下拉刷新功能

//Listview下拉刷新控件  
getRefreshControl(type){  
 **return** <RefreshControl  
 refreshing={**this**.state.isRefreshing}  
 onRefresh={**this**.\_onRefresh.bind(**this**,type)}  
 tintColor="#ff0000"  
 title="Loading..."  
 titleColor="#00ff00"  
 colors={['#ff0000', '#00ff00', '#0000ff']}  
 progressBackgroundColor="#ffff00"/>;  
}

onRefresh 里面是刷新完以后的处理

//对象的引用

class MyComponent extends React.Components {

handleClick() {

// 使用原始的DOM API来聚焦输入框。

if (this.myTextInput !== null) {

this.myTextInput.focus();

}

}

render() {

// 这里ref属性是一个回调函数，

// 它在组件加载后保存组件的引用到this.myTextInput

return (

<div>

<input type="text" ref={(ref) => this.myTextInput = ref} />

<input

type="button"

value="Focus the text input"

onClick={()=>this.handleClick()}

/>

</div>

);

}

}

ReactDOM.render(

<MyComponent />,

document.getElementById('example')

);