# **简介**

1. Facebook于2015年9月15日发布React Native。
2. React Native的优势
3. 跨平台开发
4. 追求极致的用户体验: 实时热部署
5. learn once, write everywhere
6. React Native开发注意事项
7. 目前RN在iOS上仅支持iOS7以上，Android仅支持Android4.1以上版本。
8. React Native的版本更新速度很快，建议选择
9. 功能适中、交互一般、不需要特别多的系统原生支持
10. 对于部分复杂的应用，可以考虑原生+React Native混合开发

# **环境配置**

1. 环境配置教程

<http://www.w2bc.com/article/103350>

<http://blog.csdn.net/guiman/article/details/53540511>

<http://www.myexception.cn/android/2027915.html>

<http://www.lcode.org/史上最详细windows版本搭建安装react-native环境配置/>

**<http://blog.csdn.net/zhangatle/article/details/53289471>**

注意:①安卓环境变量配置如果不成功，改成绝对路径。

②安装Android Studio，而不是Android SDK。

1. 初始化项目
2. 进入进入你希望创建项目的目录后
3. react-native init AwesomeProject
4. 进入AwesomeProject文件夹 react-native start
5. 进入以下地址

<http://localhost:8081/index.android.bundle?platform=android>

1. adb devices查看连接设备
2. 进入AwesomeProject文件夹 react-native run-android
3. react-native run-ios

注释:①项目地址不能出现汉字。②第二步和第六步是关键的两步。

# **FlexBox布局**

1. 设置样式

const style = StyleSheet.create({})

注释:未设置width和height属性的组件的宽高为0。

1. 如何让view占满整个屏幕高度

给当前view设置 flex: 1;

注意:①在Web中使用flex布局，容器需要设置display:flex;ReactNative默认布局方式为flex布局，容器无需设置display:flex。②flex属性是flex-grow, flex-shrink 和 flex-basis的简写，默认值为0 1 auto。后两个属性可选。flex:1的含义为flex-grow: 1。

1. 屏幕适配之获取屏幕分辨率

var Dimensions = require('Dimensions');

var {width, height,scale} = Dimensions.get('window');

1. React Native中的Flexbox的工作原理和web上的CSS基本一致，当然也存在少许差异。首先是默认值不同：flexDirection的默认值是column而不是row，而flex也只能指定一个数字值。
2. style属性值可以是一个数组

style={[{width: this.props.width}, styles.container]}

# **Style**

1. 绝对定位

{position:'absolute', bottom:0}



# **组件**

## 4.1 Image

1. 资源地址

(1)source={require('./img/2.png')

(2)source={require('image!icon\_homepage\_map')

(3)source={{uri:'https://www.baidu.com/img/bd\_logo1.png'}}

1. 样式

style={{width:20,height:10, resizeMode: Image.resizeMode.cover}}

(1)Image.resizeMode.cover：图片居中显示，不被拉伸，超出部分截断；

(2)Image.resizeMode.contain：容器完全容纳图片，图片等比例进拉伸；

(3)Image.resizeMode.stretch：图片被拉伸适应容器大小，可能会变形。

1. 安卓APP内置图片注意事项

(1)文件夹名称drawable-xxhdpi

(2)图片名称不可以出大写字母。

(3)文件夹图片修改之后需要重新执行react-native run-android。

(4)如果失败，重启CMD，执行react-native run-android。

(5)图片格式错误会报错:

<https://segmentfault.com/q/1010000002634455>

(6)放在android\app\src\main\res中图片的名称不能有大学字母，不能有下划线。

注意:在ReactNative中，修改文件路径和修改文件名，需要重启虚拟机，重新执行react-native run-android。

## 4.2 TextInput

1. TextInput是继承自UIView，所以View的属性TextInput也能够使用，一些样式类的属性在学习的时候可以参照View的相关属性。

注释:TextInput与Web端的input不同，没有默认的背景色和边框。

1. TextInput的常见属性

(1)value 字符串型 - 文本输入的默认值

(2)onChangeText函数 - 监听用户输入的值

(3)keyboardType - 键盘类型 例如，数字键盘

(4)multiline 布尔型 - 文本是否可以输入多行。默认值为假

(5)password 布尔型 - 文本输入框是否为密码区域。默认值为假

(6)placeholder 字符串型 - 在文本输入之前的占位文字

(7)placeholderTextColor 字符串型 - 占位符字符串的文本颜色

(8)clearButtonMode 清除按钮出现在文本视图右侧的时机

(9)enum('never', 'while-editing', 'unless-editing', 'always')

注释:multiline多行时password密码不生效。

## 4.3 Touchable

1. 高亮触摸 TouchableHighlight

当手指点击按下的时候，该视图的不透明度会进行降低同时会看到相应的颜色。此外，TouchableHighlight只能进行一层嵌套，不能多层嵌套。

常用属性：

(1)activeOpacity number

设置组件在进行触摸的时候，显示的不透明度(取值在0-1之间)

(2)onHideUnderlay function 方法

当底层被隐藏的时候调用

(3)onShowUnderlay function 方法

当底层显示的时候调用

(4)style

可以设置控件的风格演示，该风格演示可以参考View组件的style

(5)underlayColor

当触摸或者点击控件的时候显示出的颜色

1. 不透明触摸 TouchableOpacity

该组件封装了响应触摸事件；当点击按下时，该组件的透明度会降低。

常用属性：

(1)activeOpacity number

设置当用户触摸的时候，组件的透明度

常用的触摸事件

(1)onPress 点击

(2)onPressIn 按下

(3)onPressOut 抬起

(4)onLongPress 长按

## 4.4 ScrollView

1. 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。
2. ScrollView必须有一个确定的高度才能正常工作

通常有两种做法

(1)直接给该ScrollView进行设置高度(不建议)；

(2)ScrollView中不要加{flex:1}。

1. ScrollView内部的其他响应者无法阻止ScrollView本身成为响应者
2. ScrollView是继承自View，所以View中所有的属性同样适用于ScrollView。
3. ScrollView适合用来显示数量不多的滚动元素。放置在ScollView中的所有组件都会被渲染，哪怕有些组件因为内容太长被挤出了屏幕外。如果你需要显示较长的滚动列表，那么应该使用功能差不多但性能更好的ListView组件。
4. 在iOS设备上contentInset与contentOffset结合可以写出吸顶效果

contentInset {top: num, left: num, bottom: num, right: num}

内容范围相对滚动视图边缘的坐标。默认为{0, 0, 0, 0}。

contentOffset PointPropType

用来手动设置初始的滚动坐标。默认值为{x: 0, y: 0}。

## 4.5 ListView

1. ListView组件用于显示一个垂直的滚动列表，其中的元素之间结构近似而仅数据不同。ListView更适于长列表数据，且元素个数可以增删。和ScrollView不同的是，ListView并不立即渲染所有元素，而是优先渲染屏幕上可见的元素。

## 4.6 TabNavigator

1. Github地址

<https://github.com/exponent/react-native-tab-navigator>

1. npm包安装

npm install react-native-tab-navigator --save

1. 使用npm安装一个包之后如何使用

(1)打开目录..\node\_modules\react-native-tab-navigator

可以看出该包封装了Tab、TabBar、TabNavigator、TabNavigatorItem四个组件。

1. 打开..\node\_modules\react-native-tab-navigator\README.md

可以看出该包的使用方式。

1. 打开react-native-tab-navigator\TabNavagatorItem.js

通过static propTypes = {}中的键值对可以看出该组件支持的属性。

static propTypes = {

renderIcon: PropTypes.func,

renderSelectedIcon: PropTypes.func,

badgeText:PropTypes.**oneOfType**(

[PropTypes.string, PropTypes.number]),

renderBadge: PropTypes.func,

title: PropTypes.string,

titleStyle: **Text.propTypes.style**,

selectedTitleStyle: Text.propTypes.style,

tabStyle: **View.propTypes.style**,

selected: PropTypes.bool,

onPress: PropTypes.func,

allowFontScaling: PropTypes.bool,

};

注意:Text.propTypes.style这种type类型。

1. 导入

import TabNavigator from 'react-native-tab-navigator';

1. 层级

<TabNavigator>

<TabNavigator.Item>

<Navigator>

Navigator导航通常嵌套在TabNavigator.Item中。



## 4.7 Navigator

1. 参考地址: <http://www.jianshu.com/p/03ddca8f1982>
2. 该组件用来实现活动页面之间的跳转，并且能够兼容Android和iOS。
3. 两个核心要素
4. navigator

一个用来管理所有界面对象的栈，人称之“路由栈”，学名“导航”。

1. route

路由栈里的每一个对象，称之为路由，用来存储每一个页面组件以及一些需要传递给对应界面值的props参数。

1. 使用Navigator组件

return (

<Navigator style = {styles.container}

initialRoute={{

name: 'HomePage'// 可选

component: HomePage

}}

configureScene={(route) => {

return Navigator.SceneConfigs.VerticalDownSwipeJump

}}

renderScene={(route, navigator) => {

return <route.component navigator={navigator} {...route} {...route.passProps}/>

}}/>

)

1. Navigater组件的三个属性
2. initialRoute: **初始路由**，也就是我们需要在navigator栈底放置的路由，因为此时也是栈顶，因此也就是初始化整个App的首页。
3. configureScene: 设置页面**切换动画**，具体的值大家可以在**node\_modules/react-native/Libraries/CustomComponents/**Navigator/NavigatorSceneConfigs.js 源库中找到。
4. renderScene: **渲染当前路由场景**，也就是渲染栈顶路由。回调方法里一个参数是当前路由route，另一个是navigator 路由栈。

* route: 路由里包含一个**必要的**component变量和一些可选的参数，而component则就是当前需要渲染的组件。其他的一些可选的参数也是以键值对的方式保存在route对象中，并在这里以属性的方式传递给对应的component组件中。
* navigator: 这是Navigator组件在本回调方法中的参数，也就是我们在前边提到的路由栈。会在每次渲染新route的时候以属性的形式传递给route中component的对象。它可以在对应的场景中做一些操作。

总结:Navigator实现页面之间的跳转，将页面抽象为路由。路由本质上是一个键值对组成的对象，该对象必须有一个component属性，属性值为该路由由哪一个组件渲染。initialRoute初始化路由渲染成初始页。configureScene表示跳转动画。renderScene表示路由渲染成动画的函数，在该函数中可以看出路由中的数据是如何通过props进行传递的。

1. Navigator 路由栈API
2. getCurrentRoutes() - 获取当前栈里的路由，也就是push进来，没有pop掉的那些。
3. jumpBack() - 跳回之前的路由，当然前提是保留现在的，还可以再跳回来，会给你保留原样。
4. jumpForward() - 上一个方法不是调到之前的路由了么，用这个跳回来就好了。
5. jumpTo(route) - 跳转到已有的场景并且不卸载。
6. push(route) - 跳转到新的场景，并且将场景入栈，你可以稍后跳转过去。
7. pop() - 跳转回去并且卸载掉当前场景。
8. **replace(route)** - 用一个新的路由替换掉当前场景。

注意:可以使用TabNavigator替换当前Navigator。

1. replaceAtIndex(route, index) - 替换掉指定序列的路由场景。
2. replacePrevious(route) - 替换掉之前的场景。
3. immediatelyResetRouteStack(routeStack) - 用新的路由数组来重置路由栈。
4. popToRoute(route) - pop到路由指定的场景，其他的场景将会卸载。
5. popToTop() - pop到栈中的第一个场景，卸载掉所有的其他场景。

## 4.8 Platform

1. 针对不同系统做适配。
2. 导入

import {Platform} from 'react-native';

1. 使用方法

iconStyle: {

width: Platform.OS === 'ios'? 30 : 25,

height:Platform.OS === 'ios'? 30 : 25

}

## 4.9 Switch

1. 跨平台通用的可以在两个状态中切换的组件。

# **总结**

1. 使用npm安装的包在node\_modules中都可以找到

通过import XXX from '包名字'; 就可以导入，无需写具体路径。

1. 一旦调用了this.setState方法，组件一定会调用render方法，对组件进行再次的渲染，不过，React框架会自动根据diff算法来判断是否需要真正的渲染真实DOM节点。
2. style属性可以是一个普通的JavaScript对象。这是最简单的用法，因而在示例代码中很常见。你还可以传入一个数组——在数组中位置居后的样式对象比居前的优先级更高，这样你可以间接实现样式的继承。
3. isMounted此方法已经弃用很久了，主要的原因是它经过实际使用与测试可能不足以检测组件是否挂载，尤其是对于有一些异步的程序情况，以及逻辑上造成混乱。setState本身可以提供一些错误的检查，不需要这个isMounted先作检查。参考:<https://github.com/facebook/react/issues/2787>

替换写法:

constructor(props){

super(props)

this.mounted = false;

this.state={}

}

componentDidMount() {

this.mounted = true;

}

componentWillUnmount() {

this.mounted = false;

}

1. 注意这种ES6写法

this.state.text.split(' ').map((word) => word && "🍕").join(' ')

# **问题**

1. 下面两种写法的区别

this.setState({count: this.state.count + 1}) //正确写法

this.setState({count: this.state.count ++ }) //错误写法