# React Native入门教程

## React Native简介

React Native是Facebook于2015年9月发布的移动端跨平台开发解决方案，是一个相对较为成熟的跨平台开发解决方案，React Native已经完成了对多端的支持，实现了真正意义上的面向配置开发，结合React语法构建组件，开发人员可以只使用JavaScript也能编写原生移动应用，实现：Android、 iOS 使用同一套RN代码，最终编译为原生应用。

* 中文官网：<https://reactnative.cn/>
* 英文官网：<https://react-native.org/>

目前主流的跨平台开发方案有：

1. React Native(推荐)
2. Vue+Weex
3. AppCan
4. Flutter

## 原生App与webApp

React Native产出的并不是“网页应用”，并不是“HTML5应用”，并不是“混合应用”，而是一个真正的**原生移动应用**，从使用感受上和用Objective-C、Swift开发出来的IOS应用、或用Java编写的Android应用相比几乎完全一样。

### 功能

原生APP是一个系统性的应用程序，而Web APP更多是页面展示类的APP，原生APP可以调用手机终端的硬件设备（语音、摄像头、短信、GPS、蓝牙、重力感应等），Web APP则可能会受到一些限制。

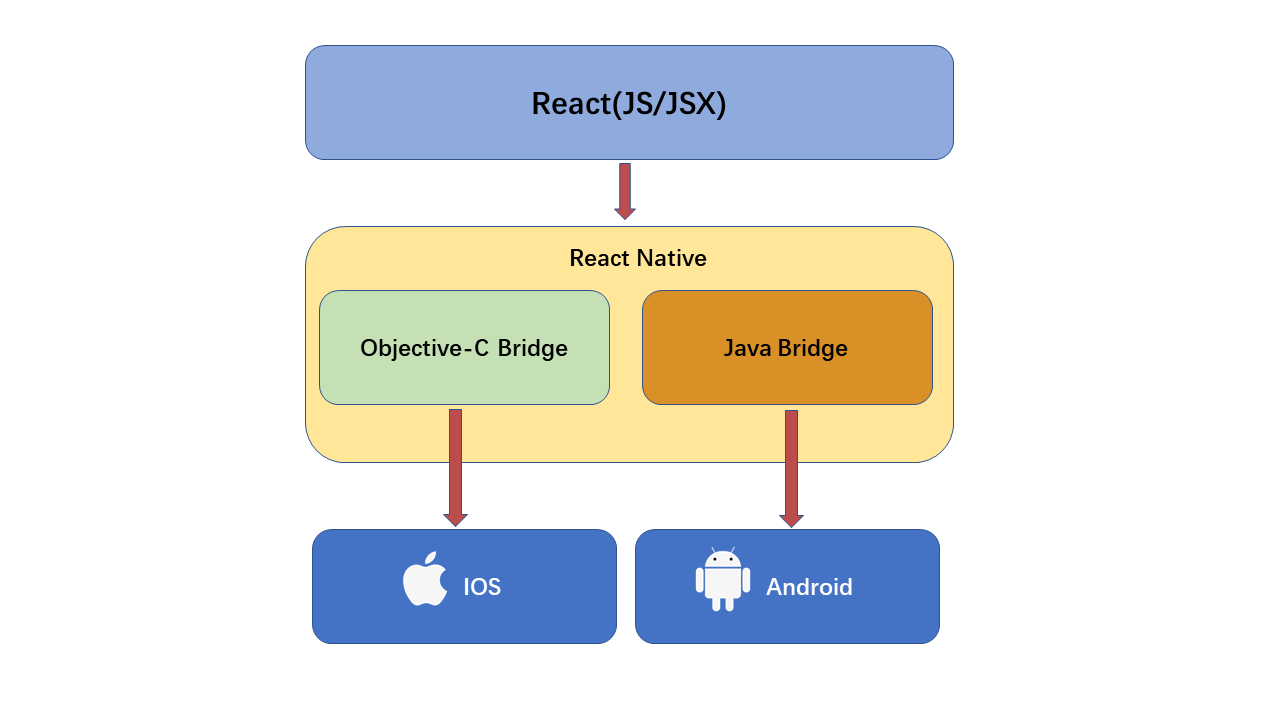
### 加载速度

原生APP所有的UI元素、数据内容、逻辑框架均安装在手机终端上。访问的时候，不需要重新下载加载，Web APP每打开一个页面，都需重新加载，访问速度受手机终端上网的限制，每次使用均会消耗一定的手机上网流量。

### 稳定性

WebbApp多为套用加壳模板，随着市场上浏览器、技术的进步，会逐步出现各种瓶颈问题，功能拓展性相对较差，而原生的APP技术更加成熟，有专业的开发环境、专门的开发语言、专门的编译环境。

### ReactNative原理图



## React Native的优势

**3.1 跨平台开发**

运用React Native，我们可以使用同一份业务逻辑核心代码来创建原生应用运行在Android端和iOS端。

**3.2追求极致的用户体验**：

热更新和热部署

**3.3 learn once，write everywhere（最具魅力）**

React Native不强求一份原生代码支持多个平台，所以不是“Write once, run anywhere”而是“Learn once, write anywhere”。

## React Native学习建议

React Native所需要的技术栈比较多，需要良好的JavaScript功底，良好的React功底，而且还需要懂一些iOS和Android原生开发，才能很好驾驭RN这个技术栈。作为初学者使用RN开发项目，建议先编写：交互简单，业务逻辑简单的应用，且尽量减少调用有兼容性的API。

## 开发环境搭建

参考react-native开发环境配置.docx

备注：若本地RN环境搭建失败可以使用在线RN编译器，地址如下：

http://dabbott.github.io/react-native-web-player/

## React Native中的特定规则

1. 样式可以直接写style={{}}，或借助特定的API创建样式，不能用传统的css或less文件来写。
2. React Native中已经默认开启flex布局，且主轴默认是纵向的。
3. React Native中依然使用jsx语法，但是不允许编写html标签，要写组件标签。

## 常用组件、常用库

### StatusBar

1. **简介：**

控制应用状态栏的组件，由于StatusBar可以在任意视图中加载，可以放置多个且后加载的会覆盖先加载的。因此在配合导航器使用时，请务必考虑清楚StatusBar的放置顺序。

1. **常用属性：**

backgroundColor：状态栏背景色

barStyle：状态栏文本颜色

hidden:是否隐藏状态栏

animated：指定状态栏的变化是否应以动画形式呈现

translucent：沉浸式状态栏

1. **基本用法：**

<View>

<StatusBar backgroundColor="blue" barStyle="light-content" />

</View>

1. **备注：npm install @types/react-native可以让vscode编码时有更好的提示。**

### SafeAreaView

1. **简介：**

SafeAreaView的目的是在一个“安全”的可视区域内渲染内容。具体来说就是因为目前有 iPhone X 这样的带有“刘海”的全面屏设备，所以需要避免内容渲染到不可见的“刘海”范围内。特别注意：本组件目前仅支持 iOS 设备以及 iOS 11 或更高版本，SafeAreaView会自动根据系统的各种导航栏、工具栏等预留出空间来渲染内部内容。更重要的是，它还会考虑到设备屏幕的局限，比如屏幕四周的圆角或是顶部中间不可显示的“刘海”区域。

1. **基本用法：**

<SafeAreaView style={{flex: 1, backgroundColor: '#fff'}}>

<View style={{ flex: 1 }}>

<Text>Hello World!</Text>

</View>

</SafeAreaView>

### View

1. **简介：**

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、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，无论它是 UIView还是 android.view.View

1. **基本用法：**

<View style={{flex: 1, backgroundColor: 'skyblue'}}>

<View style={styles.view}>

<Text style={{color: '#fff', fontSize: 18}}>内容区域</Text>

</View>

</View>

### Text

1. **简介：**

一个用于显示文本的ReactNative组件，和Android中的TextView组件或者OC中的UILabel组件相类似，专门用来显示基本的文本信息；除了基本的显示布局之外，可以进行嵌套显示，设置样式，以及可以做事件(例如:点击)处理。

**2. 常用属性：**

1.numberOfLines(number) :

设置Text文本的行数，内容超过了行数，多余的信息不会显示

2.ellipsizeMode :

这个属性通常和的 numberOfLines 属性配合使用，表示当 Text 组件无法全部显示需要显示的字符串时如何用省略号进行修饰。取值：enum('head', 'middle', 'tail', 'clip')

3.onPress (fcuntion):

该方法当文本发生点击的时候调用该方法

### Dimensions、Platform

1. **简介：**

Dimensions本模块用于获取设备屏幕的宽、高

Platform用于获取当前运行平台

1. **基本用法：**

Dimensions.get('window').width //获取宽度

Dimensions.get('window').height //获取高度

Platform.OS //获取当前运行平台

### Image、ImageBackground

1. **简介：**

用于显示多种不同类型图片的 React 组件，包括网络图片、静态资源、临时的本地图片、以及本地磁盘上的图片（如相册）等。ImageBackground与Image用法一致，区别在于ImageBackground中还可以包裹其他元素，备注：使用uri的方式引入图片，必须指定宽高，否则无法显示

1. **基本用法：**

<View>

{/\*本地加载\*/}

<Image

source={require('./images/demo.jpg')}

style={{width: 200, height: 200}}

/>

{/\*网络加载\*/}

<Image

style={{width: 168, height: 168}}

source={{uri:'http://xxxxxxxx'}}

/>

{/\*base64加载\*/}

<Image

style={{width: 66, height: 58}} source={{uri:'data:image/png;base64,iVBORw0KGgoAAAANSUhEU……'}}

/>

{/\*ImageBackground的用法\*/}

<ImageBackground

source={require('./assets/imgs/car.jpg')}

style={{width:300,height:300}}

//可选值：cover、contain、stretch、repeat、center

resizeMode="contain"

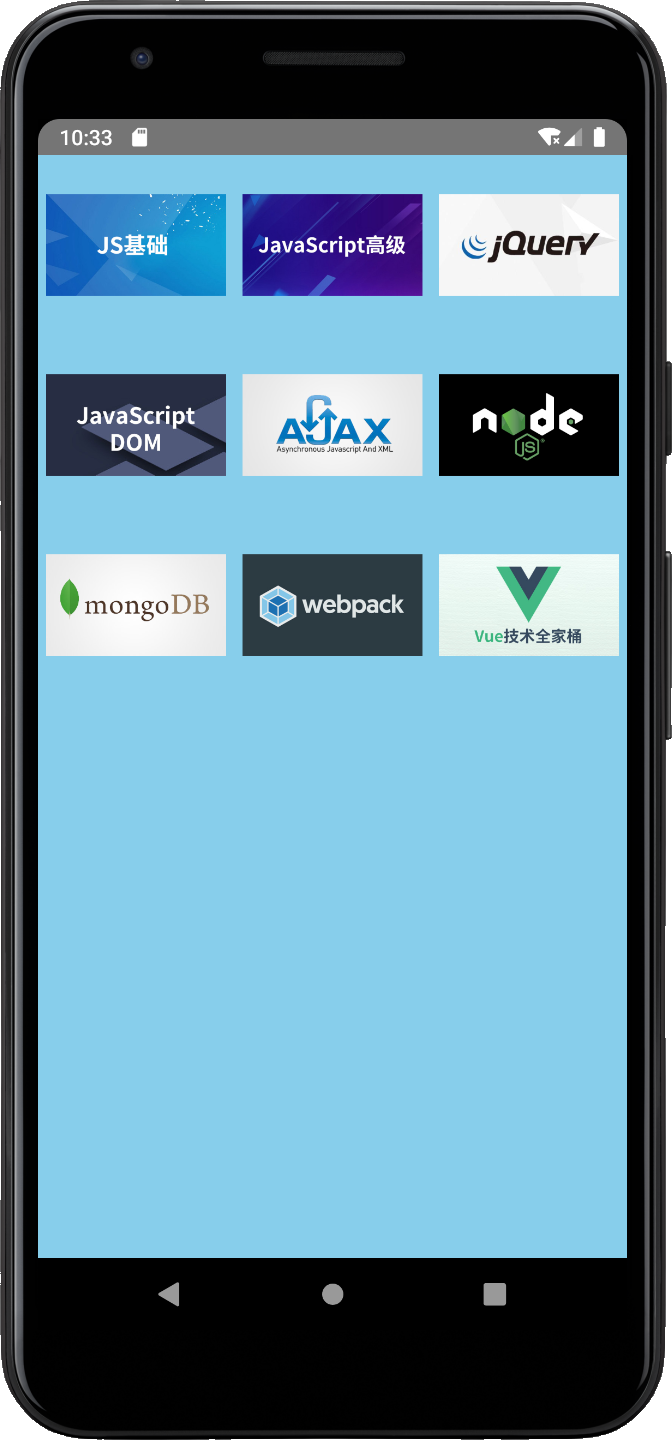
>

<Text style={{fontSize:25}}>car</Text>

</ImageBackground>

</View>

### 九宫格练习



### TextInput

1. **简介：**

TextInput是一个允许用户在应用中通过键盘输入文本的基本组件。本组件的属性提供了多种特性的配置，最简单的用法就是丢一个TextInput到应用里，然后订阅它的onChangeText事件来读取用户的输入。注意，从TextInput里取值这就是目前唯一的做法！也就是使用在onChangeText中用setState把用户的输入写入到state中，然后在需要取值的地方从this.state中取出值。

1. **基本用法：**

<TextInput

keyboardType="default" //默认键盘

placeholder="请输入数字" //提示文字

placeholderTextColor="red" //提示文字的颜色

multiline={true} //可以换行

autoCapitalize="words"//自动大写

onChangeText={(text)=>{this.setState({inputData:text})}}

style={{

height:80,

borderColor:'black',

borderWidth:1,

width:300,

marginTop:10,

marginLeft:10,

paddingLeft:10

}}

/>

### Button、TouchableOpacity

1. **简介：**

Button组件的样式是固定的。所以如果它的外观并不怎么搭配你的设计，那你需要使用TouchableOpacity组件来定制自己所需要的按钮。

1. **基本用法：**

const style = StyleSheet.create({

btnStyle:{

width:100,

height:40,

backgroundColor:'orange',

justifyContent:'center',

alignItems:'center',

borderRadius:10,

},

btnTextSttyle:{

fontSize:19,

color:'white',

}

})

<View style={style.mainStyle}>

<Button title="点我"/>

<TouchableOpacity style={style.btnStyle}>

<Text style={style.btnTextSttyle}>点我</Text>

</TouchableOpacity>

</View>

### TodoList练习



### ScrollView

1. **简介：**

ScrollView必须有一个确定的高度才能正常工作，因为它实际上所做的就是将一系列不确定高度的子组件装进一个确定高度的容器（通过滚动操作）。要给ScrollView一个确定的高度的话，要么直接给它设置高度（不建议），要么确定所有的父容器都有确定的高度。一般来说我们会给ScrollView设置flex: 1以使其自动填充父容器的空余空间，但前提条件是所有的父容器本身也设置了flex或者指定了高度，否则就会导致无法正常滚动。

**2. 常用属性：**

**1.pagingEnabled：**

**当值为true时，滚动条会停在滚动视图的尺寸的整数倍位置。**

**这个可以用在水平分页上。默认值为false。**

**2.horizontal：**

**当此属性为true的时候，所有的子视图会在水平方向上排成一行，**

**一般让horizontal为true配合pagingEnabled为true一起使用，**

**而不是默认的在垂直方向上排成一列。默认值为false**

**3.showsHorizontalScrollIndicator：是否展示滚动进度条**

**4.scrollEnabled：是否可以滚动**

1. **基本用法：**

<ScrollView

     pagingEnabled={true}

     horizontal={true}

     showsHorizontalScrollIndicator={true}

     scrollEnabled={false}

>

   {this.createColorView()}

</ScrollView>

### 下拉刷新练习



### Swiper

1. **简介：**

Swiper是一款开源、免费、强大的触摸滑动插件，对PC和移动端的兼容性良好，很多大型项目均选择该库，在RN里我们推荐使用针对react-native订制的库：react-native-swiper。

1. **Github地址：**<https://github.com/leecade/react-native-swiper>

### FlatList

1. **简介：**

当数据过多的时候ScrollView，可能会产生卡顿，且实现多列比较麻烦，使用FlatList就可以轻松实现多列，而且能实现高效的滚动，因为其内部有重用机制。特点如下：

• 完全跨平台。

• 支持水平布局模式。

• 支持单独的头部组件。

• 支持单独的尾部组件。

• 支持下拉刷新。

• 支持上拉加载。

• 支持跳转到指定行（ScrollToIndex）。

• 支持多列布局。

1. **基本用法：**

state = {

newsData:Array.from(new Array(40)).map((item,index)=>{

return {key:index,title:'标题'+index,content:'内容'+index}

})

}

<FlatList

keyExtractor={(\_, index) => index.toString()}

data={this.state.newsData}

numColumns={4}

renderItem={({item,\_})=>

<View

style={{

alignItems:'center',

width:100,

height:100,

backgroundColor:'orange'

}}

>

<Text

style={{

fontSize:18,

borderBottomWidth:1,

}}

>

{item.content}

</Text>

</View>

}

/>

1. **示例：**

****

### SectionList

1. **简介：**

高性能的分组列表组件，和FlatList相似，优势是可以方便的实现列表分组

1. **基本用法：**

<FlatList

keyExtractor={(\_, index) => index.toString()}

data={this.state.newsData}

numColumns={4}

renderItem={({item,\_})=>

<View

style={{

alignItems:'center',

width:100,

height:100,

backgroundColor:'orange'

}}

>

<Text

style={{

fontSize:18, borderBottomWidth:1,

}}

>

{item.content}

</Text>

</View>

}

/>

### SectionList练习



### 使用axios请求数据

1. **简介：**

在RN中不可避免要与后台进行数据交互，在RN中我们依然可以使用axios在ComponentDidMount这个钩子中请求后台数据

1. **基本用法：**

componentDidMount(){

axios.get('http://atguigu.utools.club/total').then(

response => {this.setState({cars:response.data})},

error => {console.log(error);}

)

}

### 集成React Navigation

1. **简介：**

在ReactNative应用中，实现多页面切换效果，主要依赖于react-navigation库实现，但并非像React一样直接引入即可，还需要一些周边库的配合。



1. **配置方法：**

**第一步：项目根路径下运行**

yarn add react-navigation

yarn add react-native-reanimated

yarn add react-native-gesture-handler

yarn add react-native-screens

yarn add react-navigation-stack

yarn add react-native-safe-area-context

yarn add @react-native-community/masked-view

**第二步：让**react-native-screens在 Android 上安装（非必须）

请在android/app/build.gradle中dependencies 选项中添加下面这两行:

implementation 'androidx.appcompat:appcompat:1.1.0-rc01'

implementation 'androidx.swiperefreshlayout:swiperefreshlayout'

1. **编码：**

第一步：将不同的“屏”拆分成不同的组件

第二步：

1. 引入createAppContainer，用于创建一个应用导航容器

import {createAppContainer} from 'react-navigation'

1. 引入createStackNavigator，用于创建栈式导航

import {createStackNavigator} from 'react-navigation-stack'

1. 创建栈式导航，并注册每一屏（类似于React中的注册路由）

//引入组件

import Home from './Home/Home'

import Detail from './Detail/Detail'

//创建一个Navigator

const Navigator = createStackNavigator({

Home:{screen:Home},

Hot:{screen:Hot},

})

1. 创建应用导航容器，并将栈式导航放入容器

const AppContainer = createAppContainer(Navigator)

export AppContainer

第三步：在App.js中渲染AppContainer

第四步：调用API屏幕切换

this.props.navigation.navigate("组件路由名字")

this.props.navigation.push("组件路由名字")

this.props.navigation.goBack()

### 导航结构、样式配置

static navigationOptions = {

title: '首页',

headerTitleStyle: {

fontWeight: 'bold',

color:'black',

textAlign:'center'

}

};

### 导航传参

**传递参数：**

this.props.navigation.navigate('路由名字',{

id:'001',

name:'atguigu'

})}

**接收参数：**

第一种方式：this.props.navigation. getParam ('参数的key')

第二种方式：this.props.navigation.state

### Tab navigation

1. **简介：**

在手机 App 中最常用的导航可能就是基于 Tab 的导航，一般都指底部导航。

1. **编码：**

第一步：yarn add react-navigation-tabs

第二步：引入createBottomTabNavigator

import {createBottomTabNavigator} from 'react-navigation-tabs'

第三步：创建Tab式导航，并注册每一屏（类似于React中的注册路由）

//引入组件

import Home from './Home/Home'

import Detail from './Detail/Detail'

//创建一个Navigator

const Navigator = createBottomTabNavigator ({

Home:{screen:Home},

Detail:{screen:Detail}

})

第四步：创建并暴露一个导航容器，并将Tab式导航放入容器

export default createAppContainer(Navigator)

### Tab navigation样式配置

Home:{

screen:Home,

navigationOptions:{

tabBarLabel:'首页',

tabBarIcon:({focused})=>

<Image

style={{width:30,height:30}}

source={{uri:focused ? 'home\_active':'home'}}

/>,

},

}

