# ReactNative

https://facebook.github.io/react-native/

移动端开发

> **WebApp**
>
> > 使用前端技术(html/css/js)，来构建出来可以运行在手机浏览器中的 application，
> >
> > 提供类似原生app的用户体验
> >
> > 优势：跨平台

> **NativeApp**
>
> > 使用原生的开发语言 (java/kotlin/oc/swift)，
> >
> > 调用系统厂商比如Google/Apple/Microsoft 所提供的SDK中的组件和服务来进行编程，
> >
> > 而生成的可以运行在手机 OS 中 app
> >
> > 优势：性能好

> **HybridApp**
>
> > 结合着web前端开发技巧和原生的开发技巧，创建可以运行在手机 os 的 app

## RN 介绍

> Build native mobile apps using JavaScript and React

**what ? when ?**

使用js 和 react 语法来构建真正的移动端的原生的app

**特点：**

Facebook 将 SDK 中的原生控件，封装成可以通过js来调用的遵循React语法的组件

**why ?**

free，较低开发成本，性能比较优秀（原生）

**开发理念：**

learn once, write anywhere

**how ?**

步骤1：安装 pc 端的工具和模版项目

npm install -g create-react-native-app

create-react-native-app myApp

cd myApp

npm start

步骤2：使用手机模拟器预览效果

① 将reactNativeForStu.zip放在C:\xampp\htdocs\framework\React

② 加压缩到当前文件夹

③ 打开vscode，打开C:\xampp\htdocs\framework\React\ReactNative\myapp

④ 在vscode打开内置终端

npm start

⑤ 检查当前的pc是否准备完毕

在浏览器中输入http://localhost:8081/

## RN 自定义组件的创建和使用

**创建:**

```
import React,{Component} from 'react';
import {Text} from 'react-native'

export default class Demo01Component extends Component{
  render(){
    return <Text>hello</Text>
  }
}
```

**调用：**

```
// index.android.js
import Demo01Component from './app/components/demo01_first'
<Demo01Component></Demo01Component>
```

## RN 常见内置组件类

- **Text**

  功能：渲染一段文本内容

  用法:

```
import {Text} from 'react-native'
<Text> 文本内容 </Text>
```

- **View**

  功能：指定一个容器

  用法:

```
import {View} from 'react-native'

<View>
  <组件/>
  <组件/>
</View>
```

- **StyleSheet**

  功能：将经常用到的样式封装对象 方便复用

  用法:

```
import {StyleSheet} from 'react-native'
var myStyles = StyleSheet.create({
  myView:{},
  myText:{color:'red',fontSize:30}
})
```

```
<Text style={myStyles.myText}></Text>
```

- **Image**

  功能：加载图片

  用法:

```
import {Image} from 'react-native'
// 情况1：加载本地图片
<Image source={require("../imgs/1.jpg")}></Image>
// 情况2：加载网络资源图片
<Image style={{width:100,height:100}} source={{uri:"http://****"}}></Image>
```

> 注意事项：
>
> ① 加载网络资源图片一定要设置宽度和高度
>
> ② 在加载本地资源图片时，不允许做任何的运算的

- **Button**

功能：实现一个按钮，点击执行操作

用法:

```
import {Button} from 'react-native'
```

```
<Button title="clickMe" onPress={this.handlePress}></Button>
```

- **state**

两个功能：

① 管理数据

```
// 初始化
constructor(){
   super();
   this.state = {
     count:1,
     myValue:2
   }
}
// 读
  this.state.count
// 写
  this.setState({count:2},()=>{})
```

② 绑定

- **FlatList**

功能： 高性能的列表组件

用法：

```
import {FlatList} from 'react-native'
showItem(info){
  return <列表项></列表项>
}
<FlatList data={****} renderItem={this.showItem}></FlatList>
```

- **TextInput**

功能：获取用户输入的信息

用法:

```
import {TextInput} from 'react-native'
<TextInput onChangeText={} value=""></TextInput>
```

```
placeholder=""
secureTextEntry=true
keyboardType:default/numeric/email-address/phone-pad
// 下面的值仅iOS可用：
ascii-capable
numbers-and-punctuation
url
number-pad
name-phone-pad
decimal-pad
twitter
web-search
```

- **受控表单元素的解决方案（状态）**

  ① 初始化状态

  ② 把状态的值绑定到视图的属性(TextInput的value)

  ③ 给受控表单元素指定一个事件处理函数，在函数中完成状态的写操作

- **Switch**

  功能：实现一个滑动开关

  用法：

```
import {Switch} from 'react-native'
```

```
<Switch><Switch>
```

> 注意事项：
>
> Switch 是一个受控的组件，需要按照受控表单元素的方式来处理

- **TouchableOpacity**

  功能：让组件内调用的组件支持按下时视图变化效果，而且支持绑定事件和事件处理函数

  用法:

```
import { TouchableOpacity } from 'react-native'
```

```
<TouchableOpacity onPress={}>
  <Image></Image>
</TouchableOpacity>
```

- **ScrollView**

  功能：实现一个支持滚动的容器

  实现:

```
import {ScrollView} from 'react-native'
```

```
<ScrollView>
 ...
</ScrollView>
```

**将一个组件固定在页面底部：**

```
<View>
  <ScrollView></ScrollView>
  <Text></Text>
</View>
```

- **FlexBox**

  功能：实现自定义布局

  - **flexDirection** : 主轴 row/column
  - **justifyContent** : 沿着主轴的对齐方式
  - **alignItems**：沿着交叉轴的对齐方式

  > 注意事项：在 rn 默认主轴是 column

  Text

  TextInput/Switch/Button

  View/ScrollView/FlatList/TouchableOpacity

  StyleSheet/state/flexbox/

  fetch

## 处理与远程服务器端的通信

### 常见的异步处理方式：

- ajax
- promise
- rxjs
- callback
- async

回顾：

```
// Angular
  HttpClient
// Vue
  axios
// React:
  fetch("")
    .then((response)=>{
        return response.json()
    })
    .then((result)=>{
        //result就是所要请求的数据
    })
```

## ReactNavigation

```
 // Vue
 // vue-router
 /*
    路由基本用法:
    ① 引入对应的路由模块
    ② 指定盛放组件的容器<router-view></router-view>
    ③ 配置路由词典、路由器
 */
var myRoutes = [
  {path:'/',component:Demo01}
]
var myRouter = new VueRouter({
  routes:myRoutes
});
new Vue({
  router:myRouter
})
```

```
// Angular
// @angular/router
// ① 安装对应的路由模块(默认官方模板项目已经指定package)
// ② 指定容器
    <router-outlet></router-outlet>
// ③ 配置路由词典
     import {Routes} from '@angular/router'
myRoutes:Routes = [{}]
 @NgModule({
  imports:[RouterModule.forRoot(myRoutes)]
 })
// ④ 测试
```

### react-navigation 概述

Routing And Navigation For React Native Apps

针对 ReactNative 的应用程序提供的路由和导航的功能

**如何使用**

**① 安装**

npm install react-navigation --save

**② 创建要用到的各个组件**

Login Register

**③ 配置路由(index.anroid.js)**

import {StackNavigator} from 'react-navigation'

import Login from '***'

import Register from '***'

```
var myNavigator = StackNavigator({
myLogin:{
  screen:Login
},
myRegister:{
  screen:Register
  }
})
```

**④ 让react-navigation接管组件的显示**

AppRegistry.registerComponent("myapp",()=>myNavigator)

## 组件之间的跳转

```
this.props.navigation.navigate("目的地的路由地址")
// 比如:
{
 myRegister:
  {screen:Demo17RegisterComponent}
}
this.props.navigation.navigate('myRegister')
```

## 组件之间跳转时完成值的传递

回顾：

```
// Vue:
// ① 明确发送方和接收方
// ② 配置接收方的路由地址
// /detail --> /detail/:id
this.$route.params.id
// ③ 发送
this.$router.push('/detail/10')
-----------------------------------------------------
// Angular:
// ① 明确发送方和接收方
// ② 配置接收方的路由地址
// /detail ---> /detail/:id
import {Route} from '@angular/router'
constructor(private myRoute:Route){}
this.myRoute.params.id
// ③ 发送
import {Router} from '@angular/router'
constructor(private myRouter:Router){}
this.myRouter.navigate('/detail/10')
```

**RN**

① 明确发送方和接收方
list-->detail
② 发送
this.props.navigation.navigate('detail',{id:1,price:20})
③ 接收
this.props.navigation.state.params.price

# 进阶知识

- **列表**

  <FlatList></FlatList>

  ① 在使用FlatList时候，如何解决key的问题？？

  ```
  data = {[1,2,3]}

  data = {[{key:0,id:1},{key:1,id:2},{}]}
  ```

> 解决方案：
>
> 在给FlatList通过data指定数据源，
>
> 保证数据源的集合中每个元素都是一个对象，必须包含key(key对应的值是不允许重复的)

② 加载更多 指定`onEndReached` `onEndReachedThreshold`

- **在RN中完成工具类的封装**

  创建：

```
// utils/global.js
export default {
  baseUrl:"http://172.163.100.193"
}
```

调用:

在任何一个组件中，引入并调用

```
import Global from '../utils/global'
```

```
Global.baseUrl
```

- **将组件固定在页面底部**

```
<View style={{flex:1}}>
  <ScrollView></ScrollView>
  <View>
    <Button></Button>
  </View>
</View>
```