# 一、建立基本开发框架（RN）

|  |
| --- |
| react-native init 项目名字 |

## 1、设置统一入口文件

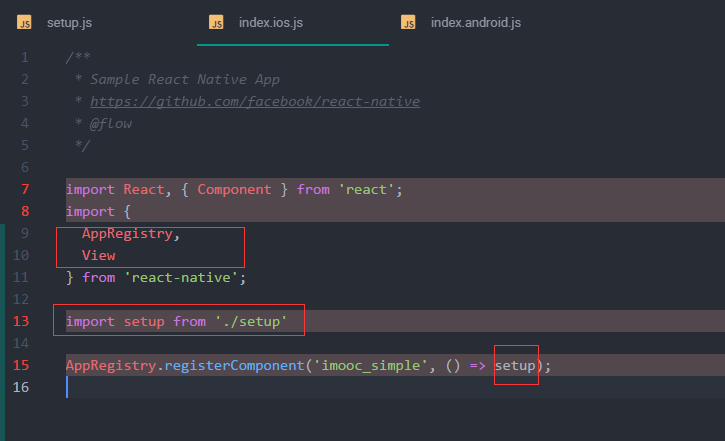
将index.ios.js和index.android.js入口文件合并成一个入口文件setup.js（文件名可自定义）

1. 将index.ios.js文件源码拷贝至setup.js文件中，修改以下内容：





1. index.ios.js与index.android.js文件内容只留存以下内容：



## 2、运行查看效果

|  |
| --- |
| 1、先打开packages管理器  react-native start ( npm start )   1. 运行在模拟器中查看效果   react-native run-android |

组件就是虚拟DOM，只有真正插入视图才是真正的DOM

# 二、相关技巧

## 1、布局相关

01、justifyContent在flexDirection:’column’时要设置父容器高度才能查看到效果，默认父容器高度是根据子元素撑开的高度决定的

02、当设置alignItems:'stretch'，flexDirection:’row’时，子组件不设置高度才会拉伸

03、当设置alignItems:'stretch'、flexDirection:’column’时，子组件不设置宽度才会拉伸

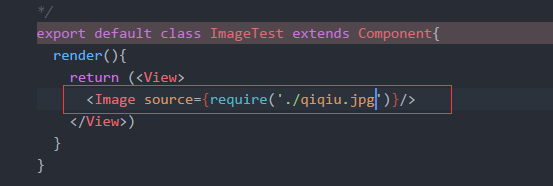
04、外边距不占据视图尺寸，内边距占据视图尺寸

05、view组件不支持设置fontSize属性（RN）

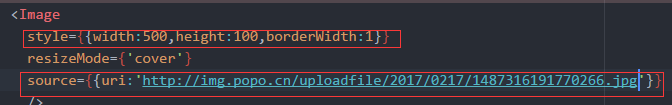
## 2、资源加载相关

### 图片加载技巧和使用详解：

1. 如何使用静态图片资源

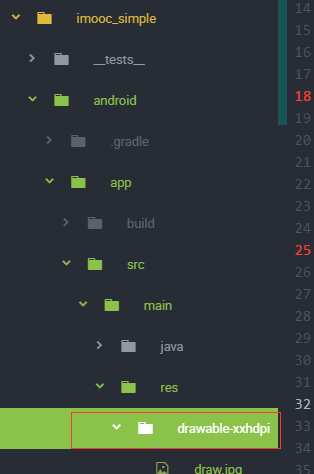


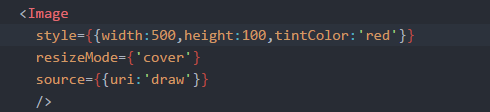
1. 如何使用网络图片资源



C、如何使用原生图片资源

在安卓项目下建立存放原生图片文件夹



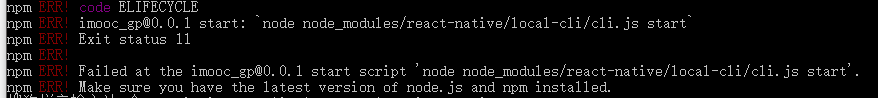


### 如何使用本地文件系统中的资源（手机相册里的资源）

### 03、图片尺寸问题

基本尺寸、2倍尺寸、3倍尺寸





### git传输http协议与ssh协议问题

将依赖包资源下载到本地，在项目中从本地安装到项目中

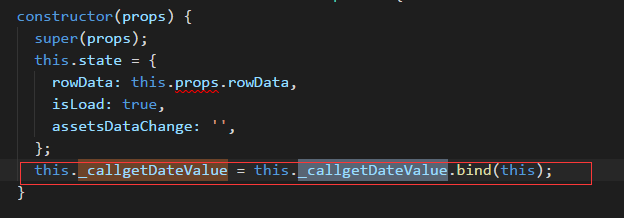
## 安装依赖

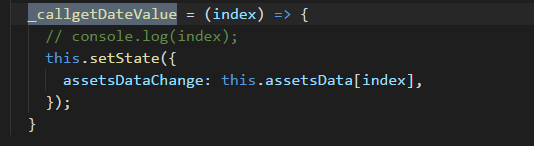
01、npm install的时候要关闭占用项目资源的软件，例如设备模拟器

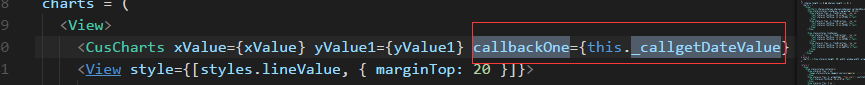
## 父子页面传参

### 01、子页面对父页面传值

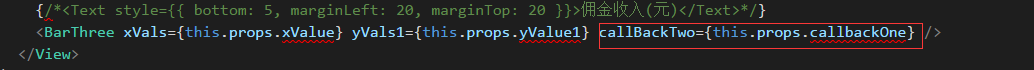
第一个页面：



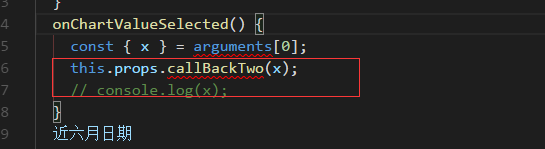




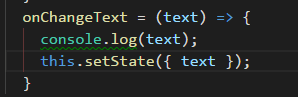
第二个页面：



第三个页面：



## 5、setState的key值和函数的参数同名写法

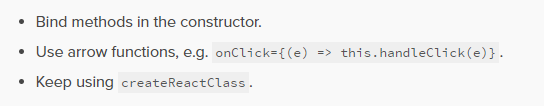


## 6、函数立即执行，箭头函数等区别

this.render 与 （）= { this.render ()} 、this.render（参数）与（）= { this.render (参数)}

this.render = this.render.bind(this);

使用函数注意事项：



## 7、react组件

### 01、react组件写法



### 02、react组件使用

React.render(myDivElement, document.body);

React.render(嵌入的组件, 嵌入的位置);



## 8、JavaScript 表达式

### 01、属性表达式



### 02、子节点表达式



### 03、注释



### 04、延展操作符



# 三、重点方法说明

## 1、遍历对象的属性

this.props.children、React.Children.map、React.Children.forEach



**Usage：**

var NotesList = React.createClass({

render: function() {

return (

<ol>

{

React.Children.map(this.props.children, function (child) {

return <li>{child}</li>;

})

}

</ol>

);

}

});

ReactDOM.render(

<NotesList>

<span>hello</span>

<span>world</span>

</NotesList>,

document.body

);

## 2、[PropTypes](https://facebook.github.io/react/docs/top-level-api.html" \l "react.proptypes)

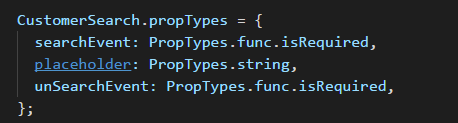
### React





|  |
| --- |
| // 可以声明prop是特定的JS基本类型  // 默认情况下这些prop都是可选的  optionalArray: React.PropTypes.array,  optionalBool: React.PropTypes.bool,  optionalFunc: React.PropTypes.func,  optionalNumber: React.PropTypes.number,  optionalObject: React.PropTypes.object,  optionalString: React.PropTypes.string,  optionalSymbol: React.PropTypes.symbol,  // 任何可以被渲染的事物：numbers, strings, elements or an array  // (or fragment) containing these types.  optionalNode: React.PropTypes.node,  // A React element.  optionalElement: React.PropTypes.element,  // 声明一个prop是某个类的实例，用到了JS的instanceof运算符  optionalMessage: React.PropTypes.instanceOf(Message),  // 用enum来限制prop只接受特定的值  optionalEnum: React.PropTypes.oneOf(['News', 'Photos']),  // 指定的多个对象类型中的一个  optionalUnion: React.PropTypes.oneOfType([  React.PropTypes.string,  React.PropTypes.number,  React.PropTypes.instanceOf(Message)  ]),  // 指定类型组成的数组  optionalArrayOf: React.PropTypes.arrayOf(React.PropTypes.number),  // 指定类型的属性构成的对象  optionalObjectOf: React.PropTypes.objectOf(React.PropTypes.number),  // 一个指定形式的对象  optionalObjectWithShape: React.PropTypes.shape({  color: React.PropTypes.string,  fontSize: React.PropTypes.number  }),  // 你可以用以上任何验证器链接‘isRequired’，来确保prop不为空  requiredFunc: React.PropTypes.func.isRequired, |

### React-native



## 3、样式

### 01、相同组件，不同样式

在父组件中通过propTypes定义style和elementStyle 强制为样式属性



### 样式覆盖原则，有值原则



数组右侧的样式权限比前面的优先，表达式属性的样式需要需要写到数组最后。

## 4、创建组件的几种方式

### 01、JavaScript function方式



### 02、ES6方式



## 5、props

### 01、调用（组件使用页）的属性



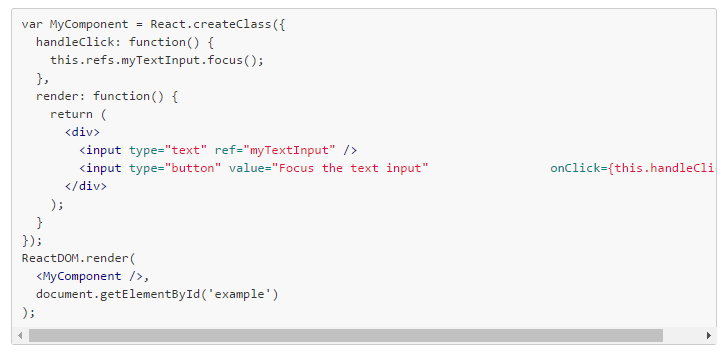
### 02、props与state

每个组件只会根据props 渲染了自己一次，props 是不可变的。

当 state 更新之后，组件就会重新渲染自己。

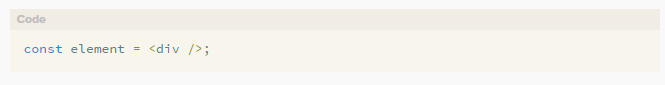
*this.props 表示那些一旦定义，就不再改变的特性，而 this.state 是会随着用户互动而产生变化的特性。*

## 6、ref 属性(获取真实的DOM节点)



## 7、其他

### 01、单组件或者没有子组件的组件需要自闭和处理



# 四、组件生命周期

## 1、组件的生命周期分成三个状态：

* Mounting：已插入真实 DOM
* Updating：正在被重新渲染
* Unmounting：已移出真实 DOM

### Mounting(装载)

* getInitialState(): 在组件挂载之前调用一次。返回值将会作为 this.state 的初始值。
* componentWillMount()：服务器端和客户端都只调用一次，在初始化渲染执行之前立刻调用。
* componentDidMount()：在初始化渲染执行之后立刻调用一次，仅客户端有效（服务器端不会调用）。

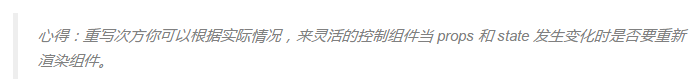
### Updating (更新)

* componentWillReceiveProps(object nextProps) 在组件接收到新的 props 的时候调用。在初始化渲染的时候，该方法不会调用。

用此函数可以作为 react 在 prop 传入之后， render() 渲染之前更新 state 的机会。老的 props 可以通过 this.props 获取到。在该函数中调用 this.setState() 将不会引起第二次渲染。

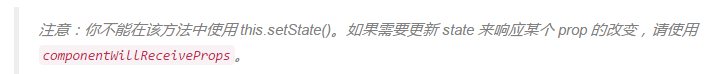
* shouldComponentUpdate(object nextProps, object nextState): 在接收到新的 props 或者 state，将要渲染之前调用。

该方法在初始化渲染的时候不会调用，在使用 forceUpdate 方法的时候也不会。如果确定新的 props 和 state 不会导致组件更新，则此处应该 返回 false。



* componentWillUpdate(object nextProps, object nextState)：在接收到新的 props 或者 state 之前立刻调用。

在初始化渲染的时候该方法不会被调用。使用该方法做一些更新之前的准备工作。



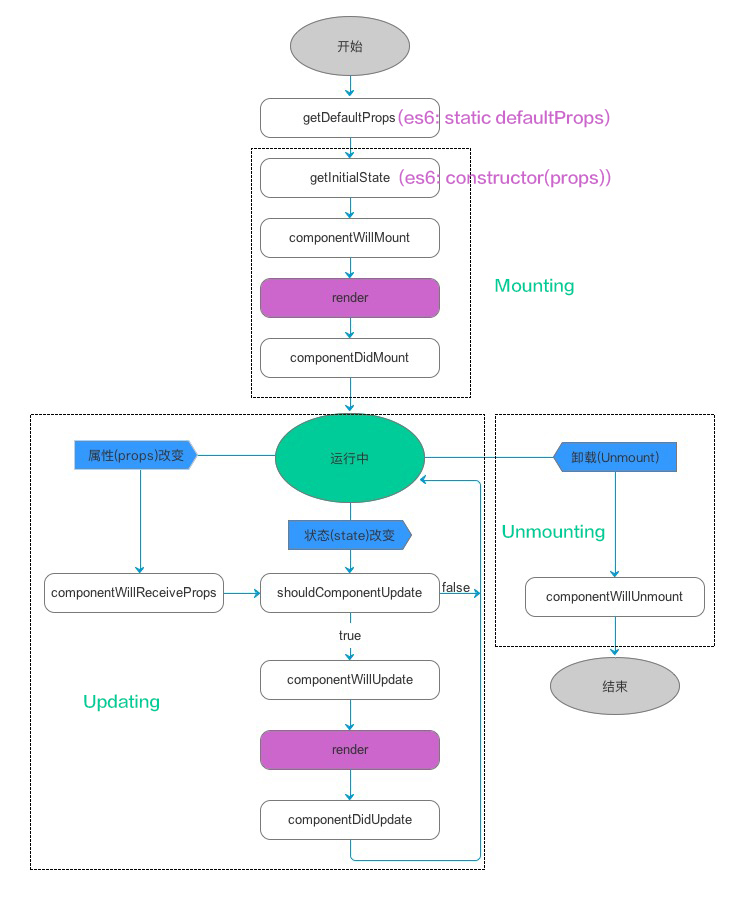
* componentDidUpdate(object prevProps, object prevState): 在组件的更新已经同步到 DOM 中之后立刻被调用。

该方法不会在初始化渲染的时候调用。使用该方法可以在组件更新之后操作 DOM 元素。

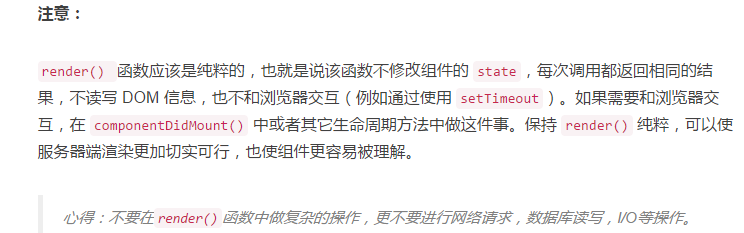
### Unmounting(移除)

* componentWillUnmount：在组件从 DOM 中移除的时候立刻被调用。

在该方法中执行任何必要的清理，比如无效的定时器，或者清除在 componentDidMount 中创建的 DOM 元素。

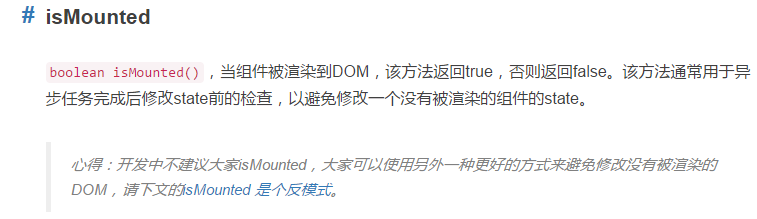


## 2、保持render的纯粹



# 五、其他

## 1、如何判断组件是否被销毁





## 2、静态方法与变量



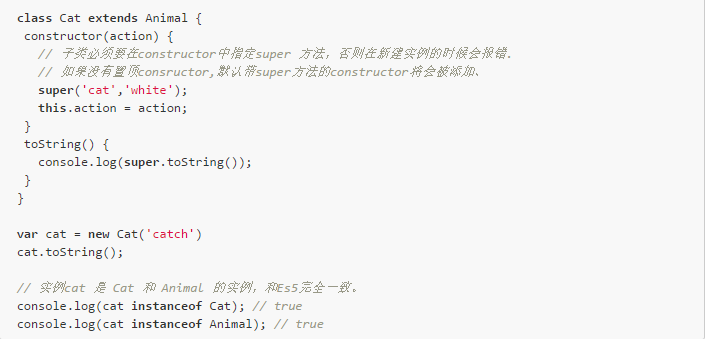
## 3、异步数据处理与本地数据存储



# 六、ES6

## 1、类（class）

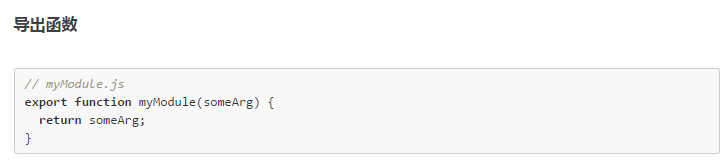




## 2、模块

### 01、导出(export)





### 02、导入(import)





## 3、箭头（Arrow）函数

### 01、箭头函数的结构



### 02、卸载监听器时的陷阱





### Mixins

增强组件来替代Mixins

### 04、ES6不再有自动绑定



### static关键字



## 4、ES6 VS ES5（ES6与ES5的区别）

### 01.在定义方面的不同

#### #定义组件



#### #定义方法



#### #定义组件的属性类型和默认属性







### 02、在导入(import)与导出(export)组件上的不同

#### #导入组件





#### #导出组件



### 03、在初始化state上的不同





### 04、在方法作为回调上的不同



### 