# 基础介绍

* 什么是React组件

Rexact组件让你将UI分割成独立、可重用的的一些碎片或者部分，这些部分都是相互独立的。

* 创建组件的三种方式

1. ES6
2. ES5
3. 函数定义

* 什么是组件的生命周期

从组件创建到消亡之间，将会经历很多的时期，这些所有时期加起来组成了所有的生命周期。

* 包含什么方法，作用

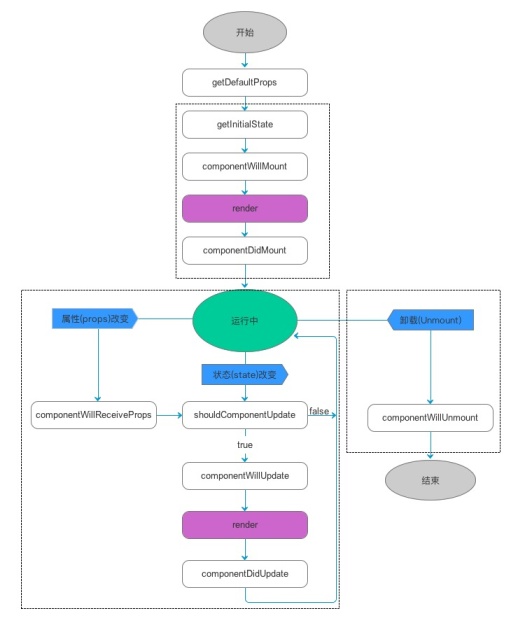


图1-1 RN 生命周期

第一阶段：是组件第一次绘制阶段，在这里完成了组件的加载和初始化；

第二阶段：是组件在运行和交互阶段或者接收事件更新界面；

第三阶段：是组件卸载消亡的阶段，组件的清理工作。

* 时期调用

# 组件使用

1. 空项目

根据之前的方式初始化空项目

1. 创建组件

创建的是js文件，名字自己写吧。

**import** React, { Component } **from** 'react';  
**import** {  
 Text,  
} **from** 'react-native';  
  
**export default class** Hello **extends** Component{  
 render(){  
 **return** <Text style={{fontSize:20}}>hello world</Text>  
 }  
}

能看得懂吧，组件继承于Component父类组件。返回样式必须要写render（）方法，返回的是界面显示的东西，我这边返回的是一个text 类型的风格为字体20的“hello world”；使用Text需要导入包。还有一点，需要导出该组件，就是用export default，其他地方就可以使用了。

三种方式：

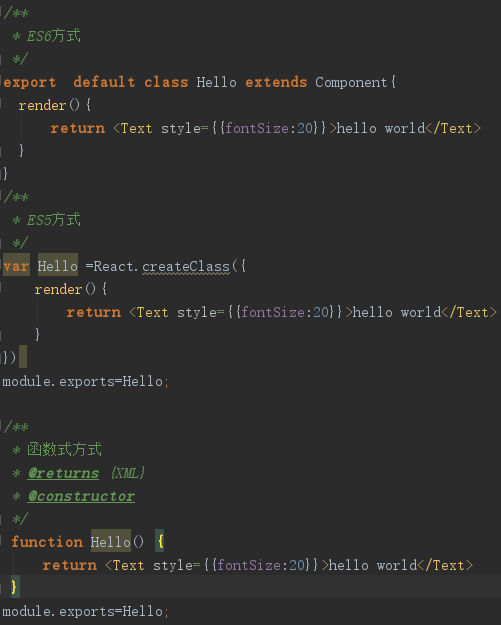


图2-1 三种组件创建方式

Module.exports=Hello 是该组件的导出方式。

1. 使用组件

首先找到主页js，使用空项目，有个index.js文件，它指向的js文件就作为开始页面。

1. 导入组件

**import** Hello **from** "./Hello";

1. 放入界面

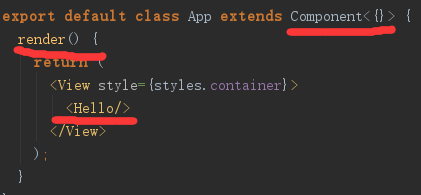


图2-2 主页入口

同样继承于Component,同样render负责所有的样式，之前创建的Hello组件放进去了。

注意：三种方式中，函数式的得到组件属性方式不太一样。

函数式：

需要添加参数。

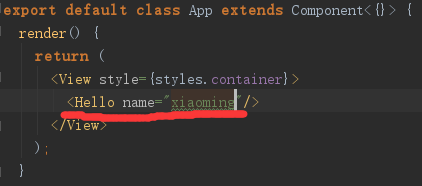


图2-3 添加属性

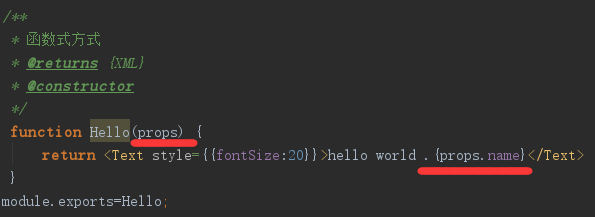


图2-4 函数式使用属性

其他方式：

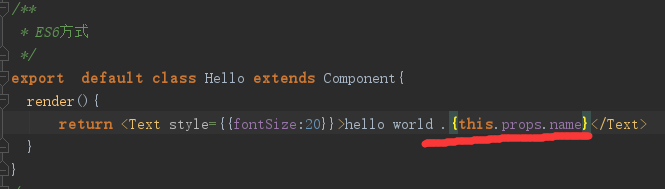


图2-5 es方式使用属性

建议：还是使用最新级的方式

# 生命周期

如果学过android或者ios，同样存在生命周期的，提供不同的函数，展示了一个界面从创建到销毁的一生。

RN中组件Component也有自己的生命方法的，具体参照图1。

下文详细介绍一下各种方法的作用。

1. 第一阶段

* 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()：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）。

1. 第二阶段

* 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。返回false 会重新渲染

初始化渲染的时候不会调用的，使用forceUpdate 方法也不会。如果props 或者 state不会导致组件更新，可以返回false。

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

注意：你不能在该方法中使用 this.setState()。如果需要更新 state 来响应某个 prop 的改变，请使用componentWillReceiveProps

* 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。

1. 第三阶段

* 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。

实例化看一下：

创建一个LifecycleComponent组件，所有生命周期的方法实现，如图：



图3-1 生命周期方法

为了能看到页面的刷新，加入了点击text，更改text样式的功能，直接看代码: 

图3-2 render代码

图中count是在state中的一个属性，在constuctor中创建：



图3-3 count创建

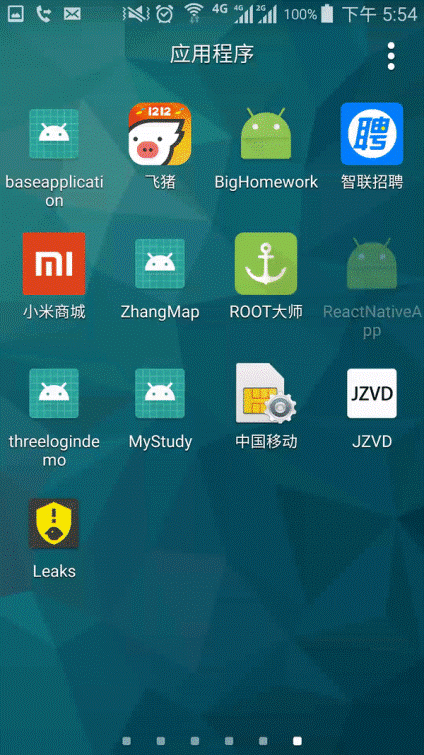


图3-4 效果

为了能看的更加直观，在每个方法中console打印除日志。

例如：

console.log('---------->componentWillMount');

在首次初始化的时候，如图：

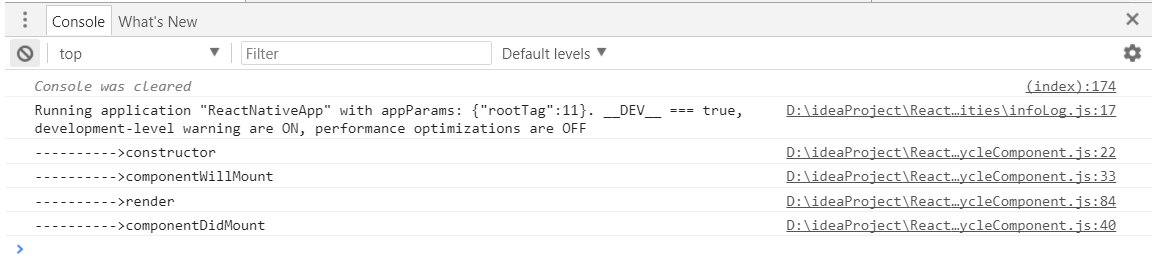


图3-5 首次初始化

点击text的时候，如图：

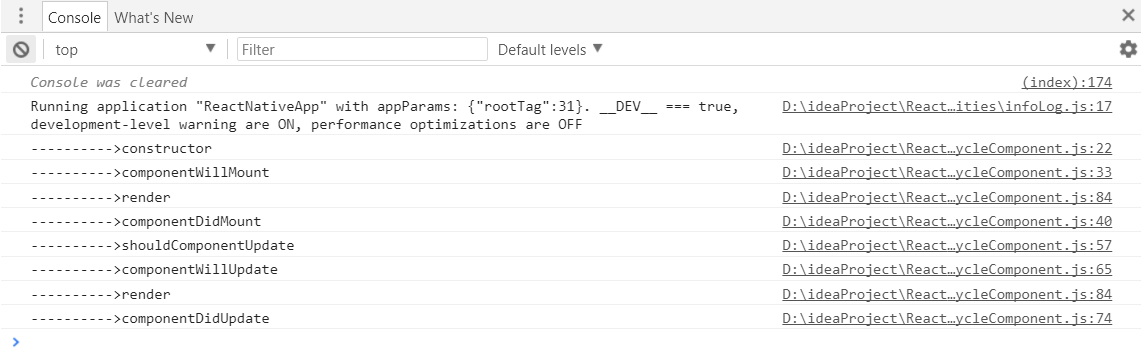


图3-6 再次渲染

装载阶段：constructor componentWillMount render componentDidMount。

更新阶段：shouldComponentUpdate componentWillUpdate render componentDidUpdate。

顺便提一下，发现webStorm这个IDE 是和intellJ idea是同源的，也很好用

注册时，在打开的License Activation窗口中选择“License server”，在输入框输入下面的网址：http://idea.codebeta.cn