概述

就像 Android 开发中的 View 一样，React Native（RN） 中的组件也有生命周期（Lifecycle）。所谓生命周期，就是一个对象从开始生成到最后消亡所经历的状态，理解生命周期，是合理开发的关键。RN 组件的生命周期整理如下图

https://upload-images.jianshu.io/upload_images/2918620-24d81978c6032593.jpg?imageMogr2/auto-orient/strip%7CimageView2/2/w/740/format/webp

如图，可以把组件生命周期大致分为三个阶段：

- 第一阶段：是组件第一次绘制阶段，如图中的上面虚线框内，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，如图中左下角虚线框，这个阶段组件可以处理用户交互，或者接收事件更新界面；
- 第三阶段：是组件卸载消亡的阶段，如图中右下角的虚线框中，这里做一些组件的清理工作。

生命周期回调函数

下面来详细介绍生命周期中的各回调函数。

getDefaultProps

在组件创建之前，会先调用 `getDefaultProps()`，这是全局调用一次，严格地来说，这不是组件的生命周期的一部分。在组件被创建并加载候，首先调用 `getInitialState()`，来初始化组件的状态。

componentWillMount

然后，准备加载组件，会调用 `componentWillMount()`，其原型如下：

```
void componentWillMount()
```

这个函数调用时机是在组件创建，并初始化了状态之后，在第一次绘制 `render()` 之前。可以在这里做一些业务初始化操作，也可以设置组件状态。这个函数在整个生命周期中只被调用一次。

componentDidMount

在组件第一次绘制之后，会调用 `componentDidMount()`，通知组件已经加载完成。函数原型如下：

```
void componentDidMount()
```

这个函数调用的时候，其虚拟 DOM 已经构建完成，你可以在这个函数开始获取其中的元素或者子组件了。需要注意的是，RN 框架是先调用子组件的 `componentDidMount()`，然后调用父组件的函数。从这个函数开始，就可以和 JS 其他框架交互了，例如设置计时 `setTimeout` 或者 `setInterval`，或者发起网络请求。这个函数也是只被调用一次。这个函数之后，就进入了稳定运行状态，等待事件触发。

componentWillReceiveProps

如果组件收到新的属性（props），就会调用 `componentWillReceiveProps()`，其原型如下：

```
void componentWillReceiveProps(
  object nextProps
)
```

输入参数 `nextProps` 是即将被设置的属性，旧的属性还是可以通过 `this.props` 来获取。在这个回调函数里面，你可以根据属性的变化，通过调用 `this.setState()` 来更新你的组件状态，这里调用更新状态是安全的，并不会触发额外的 `render()` 调用。如下：

```
componentWillReceiveProps: function(nextProps) {
  this.setState({
    likesIncreasing: nextProps.likeCount > this.props.likeCount
  });
}
```

shouldComponentUpdate

当组件接收到新的属性和状态改变的话，都会触发调用 `shouldComponentUpdate(...)`，函数原型如下：

```
boolean shouldComponentUpdate(
  object nextProps, object nextState
)
```

输入参数 `nextProps` 和上面的 `componentWillReceiveProps` 函数一样，`nextState` 表示组件即将更新的状态值。这个函数的返回值决定是否需要更新组件，如果 `true` 表示需要更新，继续走后面的更新流程。否者，则不更新，直接进入等待状态。

默认情况下，这个函数永远返回 `true` 用来保证数据变化的时候 UI 能够同步更新。在大型项目中，你可以自己重载这个函数，通过检查变化前后属性和状态，来决定 UI 是否需要更新，能有效提高应用性能。

componentWillUpdate

如果组件状态或者属性改变，并且上面的 `shouldComponentUpdate(...)` 返回为 `true`，就会开始准更新组件，并调用 `componentWillUpdate()`，其函数原型如下：

```
void componentWillUpdate(
  object nextProps, object nextState
)
```

输入参数与 `shouldComponentUpdate` 一样，在这个回调中，可以做一些在更新界面之前要做的事情。需要特别注意的是，在这个函数里面，你就不能使用 `this.setState` 来修改状态。这个函数调用之后，就会把 `nextProps` 和 `nextState` 分别设置到 `this.props`和 `this.state` 中。紧接着这个函数，就会调用 `render()` 来更新界面了。

componentDidUpdate

调用了 `render()` 更新完成界面之后，会调用 `componentDidUpdate()` 来得到通知，其函数原型如下：

```
void componentDidUpdate(
  object prevProps, object prevState
)
```

因为到这里已经完成了属性和状态的更新了，此函数的输入参数变成了 `prevProps` 和 `prevState`。

componentWillUnmount

当组件要被从界面上移除的时候，就会调用 `componentWillUnmount()`，其函数原型如下：

```
void componentWillUnmount()
```

在这个函数中，可以做一些组件相关的清理工作，例如取消计时器、网络请求等。

总结

到这里，RN 的组件的完整的生命都介绍完了，在回头来看一下前面的图，就比较清晰了，把生命周期的回调函数总结成如下表格：

生命周期	调用次数	能否使用 setSate()
getDefaultProps	1(全局调用一次)	否
getInitialState	1	否
componentWillMount	1	是
render	>=1	否
componentDidMount	1	是
componentWillReceiveProps	>=0	是
shouldComponentUpdate	>=0	否
componentWillUpdate	>=0	否
componentDidUpdate	>=0	否
componentWillUnmount	1	否