**常用定时器**

**setTimeout, clearTimeout**

在n毫秒后执行

**setInterval, clearInterval**

每隔n毫秒执行

**setImmediate, clearImmediate**

setImmediate()方法用于中断长时间运行的操作，并在浏览器完成其他操作（如事件和显示更新）后立即运行回调函数

var immediateID = setImmediate（func，[ param1，param2，...]）;

**InteractionManager**

InteractionManager提供了runAfterInteractions方法用于执行长时间的同步任务

由于js单线程的原因，执行长时间的同步任务会延迟当前进行的动画，但使用runAfterInteractions方法则不会

InteractionManager.runAfterInteractions(() => {

    // ...需要长时间同步执行的任务...

});

InteractionManager 允许应用注册动画

const handle = InteractionManager.createInteractionHandle();

// 执行动画... (`runAfterInteractions`中的任务现在开始排队等候)

// 在动画完成之后

InteractionManager.clearInteractionHandle(handle);

// 在所有句柄都清除之后，现在开始依序执行队列中的任务

**务必在卸载组件前清除定时器！**

很多 React Native 应用发生致命错误（闪退）是与计时器有关。具体来说，是在某个组件被卸载（unmount）之后，计时器却仍然在运行。

import React, { Component } from "react";

export default class Hello extends Component {

    componentDidMount() {

        this.timer = setTimeout(() => {

            console.log("把一个定时器的引用挂在this上");

        }, 500);

    }

    componentWillUnmount() {

        // 请注意Un"m"ount的m是小写

        // 如果存在this.timer，则使用clearTimeout清空。

        // 如果你使用多个timer，那么用多个变量，或者用个数组来保存引用，然后逐个clear

        this.timer && clearTimeout(this.timer);

    }

}