# ReactNative短信验证码倒计时控件

## 功能

根据项目的需要，需要写一个自定义的控件，实现如下功能：

- 默认文字为**点击获取验证码**
- 点击后出现60秒的倒计时
- 颜色，字号可调
- 倒计时过程中，再次点击需要忽略掉
- 倒计时完成后文本恢复成**点击获取验证码**
- 再几次点击同之前

其实说了这么多，就是个最普通的验证码的功能。。。

## 效果

效果图如下：(录的图片比较一般，对付着看吧)

## 实现原理

自己封装了个控件，它内部含有一个Text控件，然后我们又写了一个timer，然后负责倒计时，然后每次都需要判断一下是否继续，然后加了一个flag字段，判断是否接受下次点击事件，当倒计时结束之后还需要将初始状态重置回去即可。

## 代码

### 控件代码

```
import React, {Component  } from 'react';
import {
    StyleSheet,
    Text,
    View,
    Image,
    TextInput,
    TouchableHighlight,
    StatusBar,
    Alert,
    AppRegistry
} from 'react-native';
import LinkRow from '../components/LinkRow';
import cStyles from '../styles/CommonStyle';

import axios from 'axios';

class MyCountTime extends Component {
    constructor(props) {
        super(props);
        let timeLeft = this.props.timeLeft > 0 ? this.props.timeLeft : 5;
        let width = this.props.width || 100;
        let height = this.props.height || 50;
        let color = this.props.color || '#42A5F5';
        let fontSize = this.props.fontSize || 30;
        let fontWeight = this.props.fontWeight || '600';
        let borderColor = this.props.borderColor || '#42A5F5';
        let borderWidth = this.props.borderWidth || 1;
        let borderRadius = this.props.borderRadius || 4;
        let backgroundColor = this.props.backgroundColor || '#42A5F5';
        let begin = 0;
        let press = this.props.press;
        this.afterEnd = this.props.afterEnd || this._afterEnd;
        this.style = this.props.style;

        this.state = {
            timeLeft: timeLeft,
            begin: begin
        };
        this.countTextStyle = {
            textAlign: 'center',
            color: '#42A5F5',
            fontSize: fontSize,
            fontWeight: fontWeight

        };
        this.countViewStyle = {
            backgroundColor: backgroundColor,
            alignItems: 'center',
            borderColor: borderColor,
            borderWidth: borderWidth,
            borderRadius: borderRadius,
            width: width,
            height: height
        }
    }

    countdownfn(timeLeft, callback, begin) {
        if (timeLeft > 0) {
            this.state.begin = 1;
            console.log("===lin===>");

            let that = this;
            let interval = setInterval(function () {
                if (that.state.timeLeft < 1) {
                    clearInterval(interval);
                    callback(that)
                } else {
                    let totalTime = that.state.timeLeft;
                    that.setState({
                        timeLeft: totalTime - 1
                    })
                }
            }, 1000)
        }
    }

    _beginCountDown() {
        if (this.state.begin === 1){
            return;
        }
        let time = this.state.timeLeft;
        console.log("===lin===> time " + time);
        let afterEnd = this.afterEnd;
        let begin = this.state.begin;
        console.log("===lin===> start " + begin);
        this.countdownfn(time, afterEnd, begin)
    }

    _afterEnd(that) {
        console.log('------------time over');
        that.setState({
            begin : 0,
            timeLeft : 5,
        })
    }

    componentDidMount() {

    }

    render() {
        return (
            <View style={{position:'absolute',top:13,right:43,height:30}}>
                <Text
                    onPress={this._beginCountDown.bind(this)}
                    style={{color: '#42A5F5', fontSize: 17,height:40 , zIndex:999}}> { this.state.beg

            </View>
        )
    }
}
```

## 应用代码

```
<MyCountTime timeLeft={5}>

</MyCountTime>
```

当然这只是，最简单的应用的代码，我们还提供了很多的自定义的属性，大家可以根据自己项目的需要，去调节这些参数。

由于最近刚开始认真的搞RN，可能有一些封装的不是最佳实践，还是希望大家多提意见，和大家一起进步吧。