# Echarts

**cnpm i echarts echarts-for-react –save**

## 1.在使用图标库时可以使用echars库，可以通过按需加载引入echarts主模块

require('echarts/lib/chart/bar柱、line、pie');

var echarts = require('echarts/lib/echarts');

## 2.通过init初始化图表，并绑定Dom对象

  var myChart = echarts.init(this.refs.div1);

## 3.配置Option配置项

## 4.最后通过setOption绘制图表

**Option配置项**

myChart.setOption（option）

 var option = {

            title: { 图表标题

                text: 'ECharts 入门示例'

            },

            xAxis: { x轴显示数据

              data: ['衬衫', '羊毛衫', '雪纺衫', '裤子', '高跟鞋', '袜子']

            },

legend:{ 显示图例

 orient:'**vertical'**  **设置图例的位置**

                right:10,

                top:20,

                data:["哈罗","摩拜","小蓝"]

            },

*// y轴的值会自动计算，只需给定type类型即可*

              yAxis: {

                type: 'value'

            }, // 这个柱状图的数据链

            series: [{ 图标的数据链

                name: '销量',

                type: 'bar',

                data: [5, 20, 36, 10, 10, 20]  **纵轴数据**

areaStyle: {  **折线图显示区域填充颜色**

                     color:"rgba(128, 128, 128, 0.5)"

                 }

            }]，

radius: ["30%","70%"], ***//图的大小***

                 center:['30%','50%'],     *//图的位置*

 tooltip:{ 鼠标划入显示提示信息

               trigger:"item", 触发类型

'item'

数据项图形触发，主要在散点图，饼图等无类目轴的图表中使用。

'axis'

坐标轴触发，主要在柱状图，折线图等会使用类目轴的图表中使用。

*formatter:'{a}<br/>{b}:{c}({d}%)'*

  //    提示框浮层内容格式器，支持字符串模板和回调函数两种形式。

        //    1. 字符串模板

        //     折线（区域）图、柱状（条形）图、K线图 : {a}（系列名称），{b}（类目值），{c}（数值）, {d}（无）

        }

## Option配置项

import React, { Component } from 'react'

export default class Bar extends Component {

    render() {

        return (

            <div>

                <h3>Bar组件</h3>

                <div *ref*="div1" *style*={{ width: "600px", height: "400px" }}></div>

            </div>

        )

    }

    componentDidMount() {

*// 按需加载，在echarts库中的lib库中引入*

*// 引入 ECharts 主模块*

        var echarts = require('echarts/lib/echarts');

*// 引入柱状图*

        require('echarts/lib/chart/bar');

*// 引入提示框和标题组件*

        require('echarts/lib/component/tooltip');

        require('echarts/lib/component/title');

        var myChart = echarts.init(this.refs.div1);

let xAxisData = [];

 let yAxisData = [];

        axios.get('/data8').then(*res* => {

            let data = *res*.data.data

            for (let i in data) {

                xAxisData.push(data[i].productName)

                yAxisData.push(data[i].sales)

            }

            this.setState({ isLoading: false })

*// 绘制图表*

        myChart.setOption({

            title: {

                text: 'ECharts 入门示例'

            },

            tooltip: {},

            xAxis: {

                data: ['衬衫', '羊毛衫', '雪纺衫', '裤子', '高跟鞋', '袜子']

            },

            yAxis: {},

            series: [{

                name: '销量',

                type: 'bar',

                data: [5, 20, 36, 10, 10, 20]

            }]

        });

    }

}

# 富文本编辑器

## 1.安装两个插件，一个是富文本编辑器，一个是将文本转换成html

cnpm i -S react-draft-wysiwyg draftjs-to-html

## 2.导入插件和css样式

import { Editor } from 'react-draft-wysiwyg';

import 'react-draft-wysiwyg/dist/react-draft-wysiwyg.css';

## 3.把文本编辑器的值绑定到state上

（在编辑器上绑定*onEditorStateChange事件，*编辑器的状态发生变化时会调用函数，可用来改变 EditorState*;*

*editorState*更新编辑器状态的属性;

onEditorStateChange=(*editorState*)=>{

        this.setState({

            editorState,

          });

    }

 <Editor

*editorState*={editorState}

*onEditorStateChange*={this.onEditorStateChange}

                />

## 4.代码—有注释具体内容

import React, { Component } from 'react'

import { Button, Card, Modal } from 'antd'

import { Editor } from 'react-draft-wysiwyg';

import 'react-draft-wysiwyg/dist/react-draft-wysiwyg.css';

import draftTojs from 'draftjs-to-html';        *//将内容转换成html*

export default class Rich extends Component {

    state={

        showRichText:false,

        editorState:''

    }

    onEditorStateChange=(*editorState*)=>{

        this.setState({

            editorState,        *//显示输入的值*

          });

    }

    handdleClearContent=()=>{

        this.setState({

            editorState:'',             *//清空内容就是将输入的值设置成'  '*

          });

    }

    handdleGetText=()=>{

        this.setState({

            showRichText:true           *//显示模态框,内容再模态框编辑*

        })

    }

    onContentStateChange=(*contentState*)=>{      *//获取内容的方法*

        this.setState({

            contentState,       *//获取输入的值*

          });

    }

    render() {

        const { editorState }=this.state

        return (

            <div>

                <Card>

                    <Button *type*="primary" *onClick*={this.handdleClearContent}>清空</Button>

                    <Button *type*="primary" *onClick*={this.handdleGetText}>获取HTML文本</Button>

                </Card>

                <Card *title*="富文本编辑器" *style*={{marginTop:10}}>

                    <Editor

*editorState*={editorState}

*onContentStateChange*={this.onContentStateChange}    *//获取内容的方法*

*onEditorStateChange*={this.onEditorStateChange}

                    />

               </Card>

               <Modal

*title*="文本框内容"

*visible*={this.state.showRichText}   *//显示或隐藏根据state值决定*

*onCancel*={()=>{

                        this.setState({

                            showRichText:false

                        })

                    }}

*footer*={null}

               >

                   {

                       draftTojs(this.state.contentState)

***//调用方法,将编辑器内容转换成文本***

                   }

               </Modal>

            </div>

        )

    }

}