React-table 使用手册

作者：sq8852161
原文链接：https://ld246.com/article/1565922556128
来源网站：链滴

React-table 使用手册

本文档适用于react-table的第6版。

本文档会进行很长一段时间的更新修改，如果有发现文档错误的，欢迎指出。

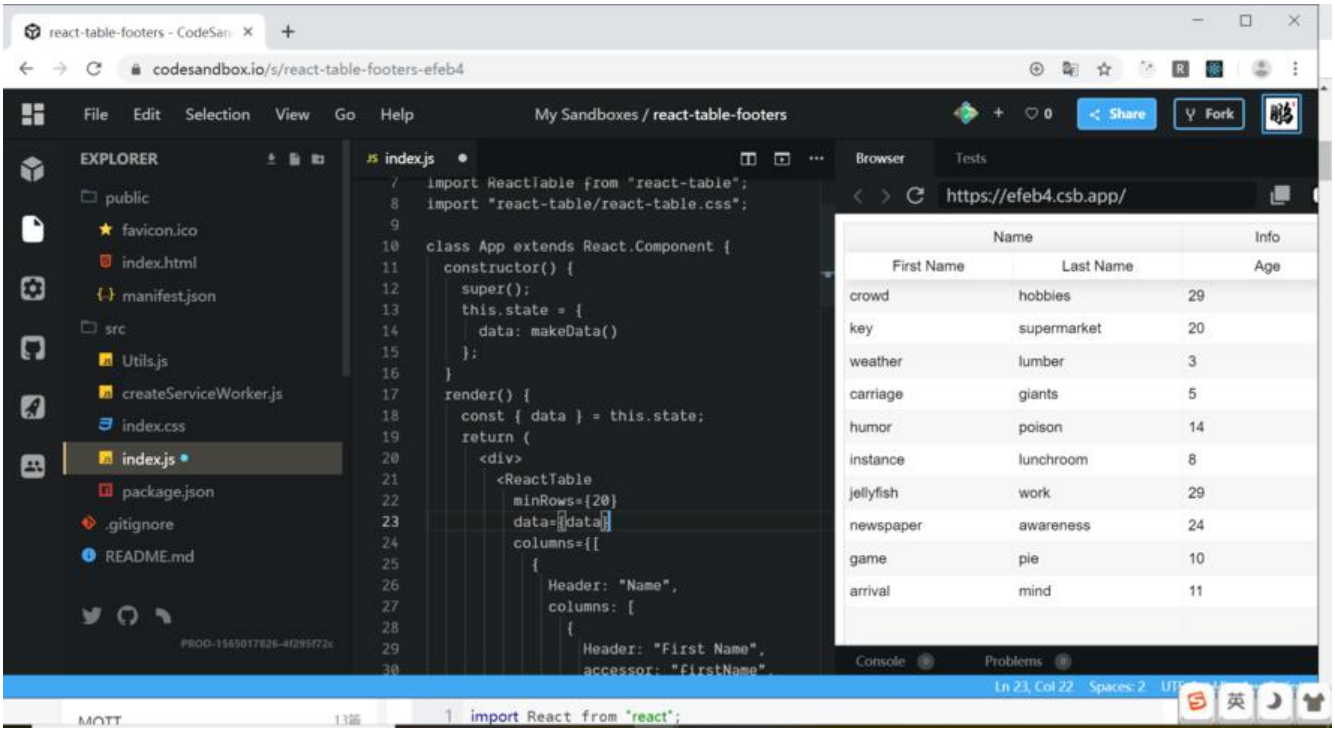
我的博客

我的邮箱:sunpeng2009@foxmail.com

react-table是一个非常好的渲染数据表的一个react的前端控件，当时之所以选择这个库就是因为其列的列宽可以自己拖拽变宽或者变窄，而且其可以非常容易的被定制。

它现在的Star已经达到了6K，npm下载量达到了26w每周。

而且其提供了可以在线练习和修改代码的范例和平台，非常的方便初学者快速掌握。



React-table的特性

- 重量轻11kb（样式仅2kb）
- 完全可定制（JSX，模板，状态，样式，回调）
- 客户端和服务器端分页
- 多重排序
- 过滤器
- 透视和聚合
- 最小的设计和易于主题化
- 通过可选的道具和回调完全可控制

安装

1，安装：

```
npm install react-table
```

2，导入react-table模块

```
// ES6
import ReactTable from 'react-table'
// ES5
var ReactTable = require('react-table').default
```

3，引入CSS

```
// JS (Webpack)
import 'react-table/react-table.css'
// Old-school
<link rel="stylesheet" href="node_modules/react-table/react-table.css">
```

如果想要改变React-table的CSS样式

1. 从npm包中的css文件开始，覆盖要更改的部分

```
import "react-table/react-table.css"
import "./your_own_styles.(s)css"
```

2. 从您自己的css文件开始，设置需要设置样式的所有内容

```
import "./your_own_styles.(s)css"
```

一个react-table.css覆盖的示例

示例

```
import ReactTable from 'react-table'

render() {
  const data = [{
    name: 'Tanner Linsley',
    age: 26,
```

```
    friend: {
      name: 'Jason Maurer',
      age: 23,
    }
  },{
    ...
  }]

  const columns = [{
    Header: 'Name',
    accessor: 'name' // String-based value accessors!
  }, {
    Header: 'Age',
    accessor: 'age',
    Cell: props => <span className='number'>{props.value}</span> // Custom cell compone
ts!
  }, {
    id: 'friendName', // Required because our accessor is not a string
    Header: 'Friend Name',
    accessor: d => d.friend.name // Custom value accessors!
  }, {
    Header: props => <span>Friend Age</span>, // Custom header components!
    accessor: 'friend.age'
  }]

  return <ReactTable
    data={data}
    columns={columns}
  />
}
```

数据

只需要传递data：数组或者对象的任何结构。react-table内置了客户端页面对数据的排序和分页，当更改props时，table会优雅的进行更新。同时，它支持服务器端数据。

```
< ReactTable
  data = { [ ... ] }
/ >
```

提示：使用resolveData来在data进行渲染前，处理data数据,从而在页面上对数据进行加工

每当dataprop值发生变化（使用===比较）时，表格都会更新，但有时您需要在数据进入表格之前其进行实现，更改或整形。要做到这一点，你可以使用resolveData道具！它接收dataprop作为唯一数并返回已解析的数据。

```
<ReactTable
  data={myData} // 数据prop应该是不可变的，只有在您想要更新表时才会更改
  resolveData={data => data.map(row => row)} //但是你可以在这里打破不变性，因为`resolveDa
a`在`data` prop改变时运行！
//此行意思应为：循环data的数据，处理每一行，第二个row，应该时处理此行数据的函数
//例子，下图为处理结果
resolveData={data =>data.map(row => {
            row.firstName = row.lastName;
```

```
                return row;
            })
        }
/>
```

	姓氏	名子
	actor	actor
	apparel	apparel
	approval	approval
	army	army
	bag	bag
	battle	battle
	battle	battle
	battle	battle
	bedroom	bedroom
	bee	bee
	berry	berry

Props

这些是主要<ReactTable />组件的所有可用Props（及其默认值）

注意，这里data:[]只是表示参数名及其默认值，实际在使用时，使用方式应为data={[]}

一般属性

data

data:[]

传输表格数据的参数，中括号内应该放需要在table中展示的“数组”或者“对象”等结构的数据

---

resolveData

resolveData: data => resolvedData　对传入的data参数进行预处理的参数接口。使用示例：

```
resolveData={data =>data.map(row => {
        row.firstName = row.lastName;
            return row;
```

```
    })
}
```

---

loading

loading:false

loading是展示否有需要loading蒙层的属性，默认false，直接改为true的话，loading蒙层就不会消了，正确用法应该是loading={判断是否使用蒙层}，{}内的值应该是可以变化的

---

showPagination

showPagination:true

是否显示分页，默认显示

---

showPaginationTop

showPaginationTop：false

是否在表格上方显示分页，默认不显示

---

showPaginationBottom

showPaginationBottom：true

是否展示底部分页，默认展示

---

showPageSizeOptions

showPageSizeOptions：ture

展示每页下拉选择每页的行数

---

pageSizeOptions

pageSizeOptions: [5, 10, 20, 25, 50, 100]

选择每页行数的参数 写法：pageSizeOptions={[22,33]}

---

defaultPageSize

defaultPageSize:20
每页行数的默认值，默认每页20行。写法：defaultPageSize={1}
这个默认数字不受pageSizeOptions的限制。

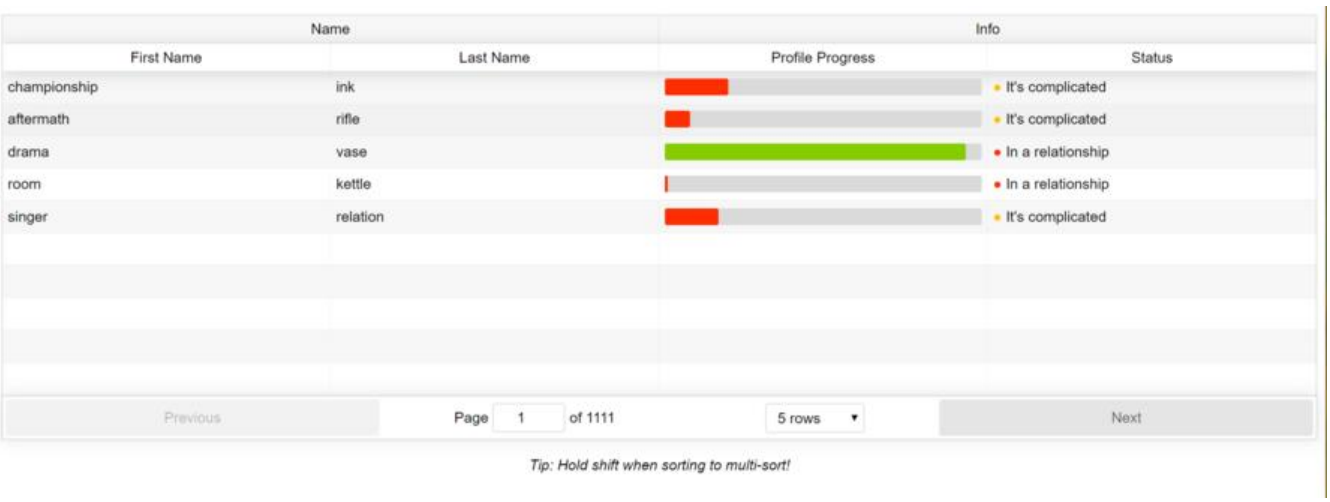
---

minRows

minRows：unfind
控制要显示的最小行数 - 默认为pageSize。这个行数，指的不是数据的行数，而是表格的行数。
如果minRows的数值小于defaultPageSize可能会出现一些奇怪的情况。
如果，这样使用：

```
defaultPageSize={5}
minRows ={10}
```

会出现如下效果



Name		Info	
First Name	Last Name	Profile Progress	Status
championship	ink		It's complicated
aftermath	rifle		It's complicated
drama	vase		In a relationship
room	kettle		In a relationship
singer	relation		It's complicated

Previous Page 1 of 1111 5 rows Next

Tip: Hold shift when sorting to multi-sort!

---

showPageJump

showPageJump：true
在分页组件上是否显示快速跳分页。默认为展示。

---

collapseOnSortingChange

---

collapseOnPageChange

---

collapseOnDataChange

---

freezeWhenExpanded

freezeWhenExpanded：false
冻结扩展时?

---

sortable

sortable：true
排序，表格列是否允许排序，默认是

---

multiSort

multiSort：true

---

resizable

resizable:true
是否可以调整大小，默认是，可以调整列的宽度

---

filterable

filterable：false
是否打开表格筛选功能，默认否

---

defaultSortDesc

defaultSortDesc：false
默认排序的正序排序还是倒序排序，false为默认正序，true为默认倒序

---

defaultSorted

defaultSorted:[]

默认排序的列，写法,其中desc为false：正序，true：倒序

```
defaultSorted={[
    {
        id: "age",
        desc: true
    }
]}
```

---

defaultFiltered

defaultFiltered:[]

默认的筛选列，及筛选值。写法

```
defaultFiltered={[
        {
            id: "age", //string 列名
            value: "29" //any 列值
            pivotId?: string //暂不清楚这个参数的作用
        }
]}
```

---

defaultResized

defaultResized:[]

设置表格默认的列的宽度。写法

```
defaultResized={[
        {
            id: "firstName",
            value: "10"
        },
        {
            id: "lastName",
            value: "1"
        },
        {
            id: "age",
            value: "1"
        }
]}
```

---

defaultExpanded

defaultExpanded：{}

默认扩展，具体写法待补充

---

defaultFilterMethod

默认列过滤方法，必须加上filterable参数。

这个方法会取代默认的排序方法。

```
defaultFilterMethod: (filter, row, column) => {
  const id = filter.pivotId || filter.id
  return row[id] !== undefined ? String(row[id]).startsWith(filter.value) : true
 },
//写法
defaultFilterMethod={(filter, row) =>
String(row[filter.id]) === filter.value}
//filter的结构
export interface Filter {
  id: string;
  value: any;
  pivotId?: string;
}
```

defaultSortMethod

默认排序方法

这个方法会取代table原本的排序方法

```
defaultSortMethod: (a, b, desc) => {
  // 强制null和undefined到底部
  a = a === null || a === undefined ? '' : a
  b = b === null || b === undefined ? '' : b
  // 强制将任何字符串值强制为小写
  a = typeof a === 'string' ? a.toLowerCase() : a
  b = typeof b === 'string' ? b.toLowerCase() : b
  // 返回1或-1表示排序优先级
  if (a > b) {
    return 1
  }
  if (a < b) {
    return -1
  }
  // 返回0，未定义或任何falsey值将使用后续排序或
  // the index as a tiebreaker
  return 0
}
```

PadRowComponent

```
PadRowComponent : ( )  =>  < span > & nbsp ; < / span > //  在填充行内呈现的内容
//实际写法
PadRowComponent={()=>
   < span > & nbsp123123 ; </span >
}// 在填充行内呈现的内容,现在不太明白使用在何处
```

受控状态覆盖（请参阅完全受控组件部分）

page

page:undefined

默认从第几页开始 page={10}

---

pageSize

pageSize:undefined

默认表格一页展示多少行数据，它会覆盖defaultPageSize的效果。

它与defaultPageSize的区别是。使用pageSize设定页面行数后，将不能在修改此表格的行数。

---

sorted

sorted:[] 表格排序的设置，与defaultSorted不同的是，使用sorted设置之后，将不能再改变排序列和排序列的正序，倒序。

```
sorted={
   [
      {id:  "age",
      desc:  true}
   ]
}
```

---

filtered

filtered: []

用法与defaultFiltered相同，但是设置好后，将不能修改过滤条件

---

resized

resized: []
用法与defaultResized相同，但是设置好后，将不能修改尺寸。

---

expanded

expanded：{}
暂不清楚用法。

受控状态回调

onPageChange

onPageChange： undefined
页面变化时回调。