zh-CN

使用受控属性对筛选和排序状态进行控制。

> 1. *columns 中定义了 filteredValue 和 sortOrder 属性即视为受控模式。*
> 2. *只支持同时对一列进行排序，请保证只有一列的 sortOrder 属性是生效的。*
> 3. *务必指定 `column.key`。*

en-US

Control filters and sorters by `filteredValue` and `sortOrder`.

> 1. *Defining `filteredValue` or `sortOrder` means that it is in the controlled mode.*
> 2. *Make sure `sortOrder` is assigned for only one column.*
> 3. *`column.key` is required.*

```
import { Table, Button, Space } from 'antd';

const data = [
  {
    key: '1',
    name: 'John Brown',
    age: 32,
    address: 'New York No. 1 Lake Park',
  },
  {
    key: '2',
    name: 'Jim Green',
    age: 42,
    address: 'London No. 1 Lake Park',
  },
  {
    key: '3',
    name: 'Joe Black',
    age: 32,
    address: 'Sidney No. 1 Lake Park',
  },
  {
    key: '4',
    name: 'Jim Red',
    age: 32,
    address: 'London No. 2 Lake Park',
  },
];

class App extends React.Component {
  state = {
    filteredInfo: null,
    sortedInfo: null,
  };
```

```
handleChange = (pagination, filters, sorter) => {
  console.log('Various parameters', pagination, filters, sorter);
  this.setState({
    filteredInfo: filters,
    sortedInfo: sorter,
  });
};

clearFilters = () => {
  this.setState({ filteredInfo: null });
};

clearAll = () => {
  this.setState({
    filteredInfo: null,
    sortedInfo: null,
  });
};

setAgeSort = () => {
  this.setState({
    sortedInfo: {
      order: 'descend',
      columnKey: 'age',
    },
  });
};

render() {
  let { sortedInfo, filteredInfo } = this.state;
  sortedInfo = sortedInfo || {};
  filteredInfo = filteredInfo || {};
  const columns = [
    {
      title: 'Name',
      dataIndex: 'name',
      key: 'name',
      filters: [
        { text: 'Joe', value: 'Joe' },
        { text: 'Jim', value: 'Jim' },
      ],
      filteredValue: filteredInfo.name || null,
      onFilter: (value, record) => record.name.includes(value),
      sorter: (a, b) => a.name.length - b.name.length,
      sortOrder: sortedInfo.columnKey === 'name' && sortedInfo.order,
      ellipsis: true,
    },
    {
      title: 'Age',
      dataIndex: 'age',
      key: 'age',
      sorter: (a, b) => a.age - b.age,
```

```
        sortOrder: sortedInfo.columnKey === 'age' && sortedInfo.order,
        ellipsis: true,
      },
      {
        title: 'Address',
        dataIndex: 'address',
        key: 'address',
        filters: [
          { text: 'London', value: 'London' },
          { text: 'New York', value: 'New York' },
        ],
        filteredValue: filteredInfo.address || null,
        onFilter: (value, record) => record.address.includes(value),
        sorter: (a, b) => a.address.length - b.address.length,
        sortOrder: sortedInfo.columnKey === 'address' && sortedInfo.order,
        ellipsis: true,
      },
    ];
    return (
      <>
        <Space style={{ marginBottom: 16 }}>
          <Button onClick={this.setAgeSort}>Sort age</Button>
          <Button onClick={this.clearFilters}>Clear filters</Button>
          <Button onClick={this.clearAll}>Clear filters and sorters</Button>
        </Space>
        <Table columns={columns} dataSource={data} onChange={this.handleChange} />
      </>
    );
  }
}

export default () => <App />;
```