zh-CN

这个例子通过简单的 ajax 读取方式，演示了如何从服务端读取并展现数据，具有筛选、排序等功能以及页面 loading 效果。开发者可以自行接入其他数据处理方式。

另外，本例也展示了筛选排序功能如何交给服务端实现，列不需要指定具体的 `onFilter` 和 `sorter` 函数，而是在把筛选和排序的参数发到服务端来处理。

当使用 `rowSelection` 时，请设置 `rowSelection.preserveSelectedRowKeys` 属性以保留 `key` 。

注意，此示例使用 [模拟接口](#)，展示数据可能不准确，请打开网络面板查看请求。

> 🛎️ *想要 3 分钟实现？试试 [ProTable](#)！*

en-US

This example shows how to fetch and present data from a remote server, and how to implement filtering and sorting in server side by sending related parameters to server.

Setting `rowSelection.preserveSelectedRowKeys` to keep the `key` when enable selection.

Note, this example use [Mock API](#) that you can look up in Network Console.

```
import { Table } from 'antd';
import qs from 'qs';

const columns = [
  {
    title: 'Name',
    dataIndex: 'name',
    sorter: true,
    render: name => `${name.first} ${name.last}`,
    width: '20%',
  },
  {
    title: 'Gender',
    dataIndex: 'gender',
    filters: [
      { text: 'Male', value: 'male' },
      { text: 'Female', value: 'female' },
    ],
    width: '20%',
  },
  {
    title: 'Email',
    dataIndex: 'email',
  },
];

const getRandomuserParams = params => ({
  results: params.pagination.pageSize,
  page: params.pagination.current,
  ...params,
});

class App extends React.Component {
  state = {
    data: [],
    pagination: {
      current: 1,
      pageSize: 10,
    },
    loading: false,
  };

  componentDidMount() {
    const { pagination } = this.state;
    this.fetch({ pagination });
  }

  handleTableChange = (pagination, filters, sorter) => {
    this.fetch({
      sortField: sorter.field,
      sortOrder: sorter.order,
      pagination,
      ...filters,
    });
  };

  fetch = (params = {}) => {
    this.setState({ loading: true });
    fetch(`https://randomuser.me/api?${qs.stringify(getRandomuserParams(params))}`)
      .then(res => res.json())
      .then(data => {
        console.log(data);
        this.setState({
          loading: false,
          data: data.results,
          pagination: {
            ...params.pagination,
            total: 200,
            // 200 is mock data, you should read it from server
            // total: data.totalCount,
          },
        });
      });
  };

  render() {
    const { data, pagination, loading } = this.state;
    return (
      <Table
        columns={columns}
        rowKey={record => record.login.uuid}
        dataSource={data}
        pagination={pagination}
        loading={loading}
        onChange={this.handleTableChange}
      />
    );
  }
}

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