zh-CN

选择不同配置组合查看效果。

en-US

Select different settings to see the result.

```
import { Table, Switch, Radio, Form, Space } from 'antd';
import { DownOutlined } from '@ant-design/icons';

const columns = [
  {
    title: 'Name',
    dataIndex: 'name',
  },
  {
    title: 'Age',
    dataIndex: 'age',
    sorter: (a, b) => a.age - b.age,
  },
  {
    title: 'Address',
    dataIndex: 'address',
    filters: [
      {
        text: 'London',
        value: 'London',
      },
      {
        text: 'New York',
        value: 'New York',
      },
    ],
    onFilter: (value, record) => record.address.indexOf(value) === 0,
  },
  {
    title: 'Action',
    key: 'action',
    sorter: true,
    render: () => (
      <Space size="middle">
        <a>Delete</a>
        <a className="ant-dropdown-link">
          More actions <DownOutlined />
        </a>
      </Space>
    ),
  },
];
```

```
const data = [];
for (let i = 1; i <= 10; i++) {
  data.push({
    key: i,
    name: 'John Brown',
    age: `${i}2`,
    address: `New York No. ${i} Lake Park`,
    description: `My name is John Brown, I am ${i}2 years old, living in New York
No. ${i} Lake Park.`,
  });
}

const expandable = { expandedRowRender: record => <p>{record.description}</p> };
const title = () => 'Here is title';
const showHeader = true;
const footer = () => 'Here is footer';
const pagination = { position: 'bottom' };

class Demo extends React.Component {
  state = {
    bordered: false,
    loading: false,
    pagination,
    size: 'default',
    expandable,
    title: undefined,
    showHeader,
    footer,
    rowSelection: {},
    scroll: undefined,
    hasData: true,
    tableLayout: undefined,
    top: 'none',
    bottom: 'bottomRight',
  };

  handleToggle = prop => enable => {
    this.setState({ [prop]: enable });
  };

  handleSizeChange = e => {
    this.setState({ size: e.target.value });
  };

  handleTableLayoutChange = e => {
    this.setState({ tableLayout: e.target.value });
  };

  handleExpandChange = enable => {
    this.setState({ expandable: enable ? expandable : undefined });
  };
```

```
  handleEllipsisChange = enable => {
    this.setState({ ellipsis: enable });
  };

  handleTitleChange = enable => {
    this.setState({ title: enable ? title : undefined });
  };

  handleHeaderChange = enable => {
    this.setState({ showHeader: enable ? showHeader : false });
  };

  handleFooterChange = enable => {
    this.setState({ footer: enable ? footer : undefined });
  };

  handleRowSelectionChange = enable => {
    this.setState({ rowSelection: enable ? {} : undefined });
  };

  handleYScrollChange = enable => {
    this.setState({ yScroll: enable });
  };

  handleXScrollChange = e => {
    this.setState({ xScroll: e.target.value });
  };

  handleDataChange = hasData => {
    this.setState({ hasData });
  };

  render() {
    const { xScroll, yScroll, ...state } = this.state;

    const scroll = {};
    if (yScroll) {
      scroll.y = 240;
    }
    if (xScroll) {
      scroll.x = '100vw';
    }

    const tableColumns = columns.map(item => ({ ...item, ellipsis: state.ellipsis
}));
    if (xScroll === 'fixed') {
      tableColumns[0].fixed = true;
      tableColumns[tableColumns.length - 1].fixed = 'right';
    }

    return (
      <>
```

```
        <Form
          layout="inline"
          className="components-table-demo-control-bar"
          style={{ marginBottom: 16 }}
        >
          <Form.Item label="Bordered">
            <Switch checked={state.bordered} onChange=
{this.handleToggle('bordered')} />
          </Form.Item>
          <Form.Item label="loading">
            <Switch checked={state.loading} onChange={this.handleToggle('loading')}
/>
          </Form.Item>
          <Form.Item label="Title">
            <Switch checked={!!state.title} onChange={this.handleTitleChange} />
          </Form.Item>
          <Form.Item label="Column Header">
            <Switch checked={!!state.showHeader} onChange={this.handleHeaderChange}
/>
          </Form.Item>
          <Form.Item label="Footer">
            <Switch checked={!!state.footer} onChange={this.handleFooterChange} />
          </Form.Item>
          <Form.Item label="Expandable">
            <Switch checked={!!state.expandable} onChange={this.handleExpandChange}
/>
          </Form.Item>
          <Form.Item label="Checkbox">
            <Switch checked={!!state.rowSelection} onChange=
{this.handleRowSelectionChange} />
          </Form.Item>
          <Form.Item label="Fixed Header">
            <Switch checked={!!yScroll} onChange={this.handleYScrollChange} />
          </Form.Item>
          <Form.Item label="Has Data">
            <Switch checked={!!state.hasData} onChange={this.handleDataChange} />
          </Form.Item>
          <Form.Item label="Ellipsis">
            <Switch checked={!!state.ellipsis} onChange={this.handleEllipsisChange}
/>
          </Form.Item>
          <Form.Item label="Size">
            <Radio.Group value={state.size} onChange={this.handleSizeChange}>
              <Radio.Button value="default">Default</Radio.Button>
              <Radio.Button value="middle">Middle</Radio.Button>
              <Radio.Button value="small">Small</Radio.Button>
            </Radio.Group>
          </Form.Item>
          <Form.Item label="Table Scroll">
            <Radio.Group value={xScroll} onChange={this.handleXScrollChange}>
              <Radio.Button value={undefined}>Unset</Radio.Button>
              <Radio.Button value="scroll">Scroll</Radio.Button>
```

```
              <Radio.Button value="fixed">Fixed Columns</Radio.Button>
            </Radio.Group>
          </Form.Item>
          <Form.Item label="Table Layout">
            <Radio.Group value={state.tableLayout} onChange=
{this.handleTableLayoutChange}>
              <Radio.Button value={undefined}>Unset</Radio.Button>
              <Radio.Button value="fixed">Fixed</Radio.Button>
            </Radio.Group>
          </Form.Item>
          <Form.Item label="Pagination Top">
            <Radio.Group
              value={this.state.top}
              onChange={e => {
                this.setState({ top: e.target.value });
              }}
            >
              <Radio.Button value="topLeft">TopLeft</Radio.Button>
              <Radio.Button value="topCenter">TopCenter</Radio.Button>
              <Radio.Button value="topRight">TopRight</Radio.Button>
              <Radio.Button value="none">None</Radio.Button>
            </Radio.Group>
          </Form.Item>
          <Form.Item label="Pagination Bottom">
            <Radio.Group
              value={this.state.bottom}
              onChange={e => {
                this.setState({ bottom: e.target.value });
              }}
            >
              <Radio.Button value="bottomLeft">BottomLeft</Radio.Button>
              <Radio.Button value="bottomCenter">BottomCenter</Radio.Button>
              <Radio.Button value="bottomRight">BottomRight</Radio.Button>
              <Radio.Button value="none">None</Radio.Button>
            </Radio.Group>
          </Form.Item>
        </Form>
        <Table
          {...this.state}
          pagination={{ position: [this.state.top, this.state.bottom] }}
          columns={tableColumns}
          dataSource={state.hasData ? data : null}
          scroll={scroll}
        />
      </>
    );
  }
}

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