Table

A table displays rows of data.

When To Use

- To display a collection of structured data.
- To sort, search, paginate, filter data.

How To Use

Specify `dataSource` of Table as an array of data.

```
const dataSource = [
  {
    key: '1',
    name: 'Mike',
    age: 32,
    address: '10 Downing Street',
  },
  {
    key: '2',
    name: 'John',
    age: 42,
    address: '10 Downing Street',
  },
];

const columns = [
  {
    title: 'Name',
    dataIndex: 'name',
    key: 'name',
  },
  {
    title: 'Age',
    dataIndex: 'age',
    key: 'age',
  },
  {
    title: 'Address',
    dataIndex: 'address',
    key: 'address',
  },
];

<Table dataSource={dataSource} columns={columns} />;
```

Promotion

- Kitchen Sketch Plugin 💎
- ProTable - Advanced Tables
- S2 - Analytical Tables

Examples

Name	Age	Address	Tags	Action
John Brown	32	New York No. 1 Lake Park	NICE DEVELOPER	Invite John Brown Delete
Jim Green	42	London No. 1 Lake Park	LOSER	Invite Jim Green Delete
Joe Black	32	Sydney No. 1 Lake Park	COOL TEACHER	Invite Joe Black Delete

< 1 >

Basic Usage

Simple table with actions.

```
import React from 'react';
import { Space, Table, Tag } from 'antd';
import type { ColumnsType } from 'antd/es/table';

interface DataType {
  key: string;
  name: string;
  age: number;
  address: string;
  tags: string[];
}

const columns: ColumnsType<DataType> = [
  {
    title: 'Name',
    dataIndex: 'name',
    key: 'name',
    render: (text) => <a>{text}</a>,
  },
  {
    title: 'Age',
    dataIndex: 'age',
    key: 'age',
  },
  {
    title: 'Address',
    dataIndex: 'address',
    key: 'address',
  },
  {
    title: 'Tags',
    key: 'tags',
    dataIndex: 'tags',
    render: (_, { tags }) => (
      <>
        {tags.map((tag) => {
          let color = tag.length > 5 ? 'geekblue' : 'green';
          if (tag === 'loser') {
            color = 'volcano';
          }
          return (
            <Tag color={color} key={tag}>
              {tag.toUpperCase()}
            </Tag>
          );
        })}
      </>
    ),
  },
  {
```

Name		Age	Address	Tags	Action	
First Name	Last Name					
John	Brown	32	New York No. 1 Lake Park	nice developer	Invite Brown	Delete
Jim	Green	42	London No. 1 Lake Park	loser	Invite Green	Delete
Joe	Black	32	Sydney No. 1 Lake Park	cool teacher	Invite Black	Delete

< 1 >

JSX style API

Using JSX style API (introduced in 2.5.0)

> Since this is just a syntax sugar for the prop `columns`, you can't compose `Column` and `ColumnGroup` with other Components.

```
import React from 'react';
import { Space, Table, Tag } from 'antd';

const { Column, ColumnGroup } = Table;

interface DataType {
  key: React.Key;
  firstName: string;
  lastName: string;
  age: number;
  address: string;
  tags: string[];
}

const data: DataType[] = [
  {
    key: '1',
    firstName: 'John',
    lastName: 'Brown',
    age: 32,
    address: 'New York No. 1 Lake Park',
    tags: ['nice', 'developer'],
  },
  {
    key: '2',
    firstName: 'Jim',
    lastName: 'Green',
    age: 42,
    address: 'London No. 1 Lake Park',
    tags: ['loser'],
  },
  {
    key: '3',
    firstName: 'Joe',
    lastName: 'Black',
    age: 32,
    address: 'Sydney No. 1 Lake Park',
    tags: ['cool', 'teacher'],
  },
];
```

Checkbox radio

Name	Age	Address
John Brown	32	New York No. 1 Lake Park
Jim Green	42	London No. 1 Lake Park
Joe Black	32	Sydney No. 1 Lake Park
Disabled User	99	Sydney No. 1 Lake Park

1

selection

Rows can be selectable by making first column as a selectable column. You can use `rowSelection.type` to set selection type. Default is `checkbox` .

> selection happens when clicking checkbox by default. You can see https://codesandbox.io/s/000vqw38rl if you need row-click selection behavior.

```
import React, { useState } from 'react';
import { Divider, Radio, Table } from 'antd';
import type { ColumnsType } from 'antd/es/table';

interface DataType {
  key: React.Key;
  name: string;
  age: number;
  address: string;
}

const columns: ColumnsType<DataType> = [
  {
    title: 'Name',
    dataIndex: 'name',
    render: (text: string) => <a>{text}</a>,
  },
  {
    title: 'Age',
    dataIndex: 'age',
  },
  {
    title: 'Address',
    dataIndex: 'address',
  },
];

const data: DataType[] = [
  {
    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',
  },
```

Reload

Name	Age	Address
Edward King 0	32	London, Park Lane no. 0
Edward King 1	32	London, Park Lane no. 1
Edward King 2	32	London, Park Lane no. 2
Edward King 3	32	London, Park Lane no. 3
Edward King 4	32	London, Park Lane no. 4
Edward King 5	32	London, Park Lane no. 5
Edward King 6	32	London, Park Lane no. 6
Edward King 7	32	London, Park Lane no. 7
Edward King 8	32	London, Park Lane no. 8
Edward King 9	32	London, Park Lane no. 9

< 1 2 3 4 5 >

Selection and operation

To perform operations and clear selections after selecting some rows, use `rowSelection.selectedRowKeys` to control selected rows.

```
import React, { useState } from 'react';
import { Button, Table } from 'antd';
import type { ColumnsType } from 'antd/es/table';

interface DataType {
  key: React.Key;
  name: string;
  age: number;
  address: string;
}

const columns: ColumnsType<DataType> = [
  {
    title: 'Name',
    dataIndex: 'name',
  },
  {
    title: 'Age',
    dataIndex: 'age',
  },
  {
    title: 'Address',
    dataIndex: 'address',
  },
];

const data: DataType[] = [];
for (let i = 0; i < 46; i++) {
```

Name	Age	Address
Edward King 0	32	London, Park Lane no. 0
Edward King 1	32	London, Park Lane no. 1
Edward King 2	32	London, Park Lane no. 2
Edward King 3	32	London, Park Lane no. 3
Edward King 4	32	London, Park Lane no. 4
Edward King 5	32	London, Park Lane no. 5
Edward King 6	32	London, Park Lane no. 6
Edward King 7	32	London, Park Lane no. 7
Edward King 8	32	London, Park Lane no. 8
Edward King 9	32	London, Park Lane no. 9

Custom selection

Use `rowSelection.selections` custom selections, default no select dropdown, show default selections via setting to `true`.

```
import React, { useState } from 'react';
import { Table } from 'antd';
import type { ColumnsType } from 'antd/es/table';
import type { TableRowSelection } from 'antd/es/table/interface';

interface DataType {
  key: React.Key;
  name: string;
  age: number;
  address: string;
}

const columns: ColumnsType<DataType> = [
  {
    title: 'Name',
    dataIndex: 'name',
  },
  {
    title: 'Age',
    dataIndex: 'age',
  },
  {
    title: 'Address',
    dataIndex: 'address',
  },
];

const data: DataType[] = [];
for (let i = 0; i < 46; i++) {
  data.push({
```

Name	Age	Address
Jim Green	42	London No. 1 Lake Park
John Brown	32	New York No. 1 Lake Park
Joe Black	32	Sydney No. 1 Lake Park
Jim Red	32	London No. 2 Lake Park

1

Filter and sorter

Use `filters` to generate filter menu in columns, `onFilter` to determine filtered result, and `filterMultiple` to indicate whether it's multiple or single selection.

Uses `defaultFilteredValue` to make a column filtered by default.

Use `sorter` to make a column sortable. `sorter` can be a function of the type `function(a, b) { ... }` for sorting data locally.

`sortDirections: ['ascend' | 'descend']` defines available sort methods for each columns, effective for all columns when set on table props. You can set as `['ascend', 'descend', 'ascend']` to prevent sorter back to default status.

Uses `defaultSortOrder` to make a column sorted by default.

If a `sortOrder` or `defaultSortOrder` is specified with the value `ascend` or `descend`, you can access this value from within the function passed to the `sorter` as explained above. Such a function can take the form: `function(a, b, sortOrder) { ... }`.

```
import React from 'react';
import { Table } from 'antd';
import type { ColumnsType, TableProps } from 'antd/es/table';

interface DataType {
  key: React.Key;
  name: string;
  age: number;
  address: string;
}

const columns: ColumnsType<DataType> = [
  {
    title: 'Name',
    dataIndex: 'name',
    filters: [
      {
        text: 'Joe',
        value: 'Joe',
      },
      {
        text: 'Jim',
        value: 'Jim',
      },
      {
        text: 'Submenu',
        value: 'Submenu',
        children: [
          {
            text: 'Green',
            value: 'Green',
          },
```

Name	Age	Address
John Brown	32	New York No. 1 Lake Park
Jim Green	42	London No. 1 Lake Park
Joe Black	32	Sydney No. 1 Lake Park
Jim Red	32	London No. 2 Lake Park

1

Filter in Tree

You can use `filterMode` to change default filter interface, options: `menu` (default) and `tree`.

> `filterSearch` is used for making filter dropdown items searchable.

```
import React from 'react';
import { Table } from 'antd';
import type { ColumnsType, TableProps } from 'antd/es/table';

interface DataType {
  key: React.Key;
  name: string;
  age: number;
  address: string;
}

const columns: ColumnsType<DataType> = [
  {
    title: 'Name',
    dataIndex: 'name',
    filters: [
      {
        text: 'Joe',
        value: 'Joe',
      },
      {
        text: 'Category 1',
        value: 'Category 1',
        children: [
          {
            text: 'Yellow',
            value: 'Yellow',
          },
          {
            text: 'Pink',
            value: 'Pink',
          },
        ],
      },
      {
        text: 'Category 2',
        value: 'Category 2',
        children: [
          {
            text: 'Green',
            value: 'Green',
          },
          {
            text: 'Black',
            value: 'Black',
          },
```

Name	Age	Address
John Brown	32	New York No. 1 Lake Park
Jim Green	42	London No. 1 Lake Park
Joe Black	32	Sydney No. 1 Lake Park
Jim Red	32	London No. 2 Lake Park

Filter search

`filterSearch` is used to enable search of filter items, and you can set a custom filter method through `filterSearch:(input, record) => boolean`.

```
import React from 'react';
import { Table } from 'antd';
import type { ColumnsType, TableProps } from 'antd/es/table';

interface DataType {
  key: React.Key;
  name: string;
  age: number;
  address: string;
}

const columns: ColumnsType<DataType> = [
  {
    title: 'Name',
    dataIndex: 'name',
    filters: [
      {
        text: 'Joe',
        value: 'Joe',
      },
      {
        text: 'Category 1',
        value: 'Category 1',
      },
      {
        text: 'Category 2',
        value: 'Category 2',
      },
    ],
    filterMode: 'tree',
    filterSearch: true,
    onFilter: (value: string, record) => record.name.startsWith(value),
    width: '30%',
  },
  {
    title: 'Age',
    dataIndex: 'age',
    sorter: (a, b) => a.age - b.age,
  },
  {
    title: 'Address',
    dataIndex: 'address',
    filters: [
      {
        text: 'London',
        value: 'London',
      },
```

Name	Chinese Score	Math Score	English Score
John Brown	98	60	70
Jim Green	98	66	89
Joe Black	98	90	70
Jim Red	88	99	89

Multiple sorter

`column.sorter` support `multiple` to config the priority of sort columns. Though `sorter.compare` to customize compare function. You can also leave it empty to use the interactive only.

```
import React from 'react';
import { Table } from 'antd';
import type { ColumnsType, TableProps } from 'antd/es/table';

interface DataType {
  key: React.Key;
  name: string;
  chinese: number;
  math: number;
  english: number;
}

const columns: ColumnsType<DataType> = [
  {
    title: 'Name',
    dataIndex: 'name',
  },
  {
    title: 'Chinese Score',
    dataIndex: 'chinese',
    sorter: {
      compare: (a, b) => a.chinese - b.chinese,
      multiple: 3,
    },
  },
  {
    title: 'Math Score',
    dataIndex: 'math',
    sorter: {
      compare: (a, b) => a.math - b.math,
      multiple: 2,
    },
  },
  {
    title: 'English Score',
    dataIndex: 'english',
    sorter: {
      compare: (a, b) => a.english - b.english,
      multiple: 1,
    },
  },
];

const data: DataType[] = [
  {
    key: '1',
    name: 'John Brown',
```

Sort age | Clear filters | Clear filters and sorters

Name	Age	Address
John Brown	32	New York No. 1 Lake Park
Jim Green	42	London No. 1 Lake Park
Joe Black	32	Sydney No. 1 Lake Park
Jim Red	32	London No. 2 Lake Park

1

Reset filters and sorters

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 React, { useState } from 'react';
import type { TableProps } from 'antd';
import { Button, Space, Table } from 'antd';
import type { ColumnsType, FilterValue, SorterResult } from 'antd/es/table/interface';

interface DataType {
  key: string;
  name: string;
  age: number;
  address: string;
}

const data: DataType[] = [
  {
    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: 'Sydney No. 1 Lake Park',
  },
  {
    key: '4',
    name: 'Jim Red',
    age: 32,
    address: 'London No. 2 Lake Park',
  },
];

const App: React.FC = () => {
  const [filteredInfo, setFilteredInfo] = useState<Record<string, FilterValue | null>>({});
  const [sortedInfo, setSortedInfo] = useState<SorterResult<DataType>>({});
```

Name	Age	Address
John Brown	32	New York No. 1 Lake Park
Joe Black	42	London No. 1 Lake Park
Jim Green	32	Sydney No. 1 Lake Park
Jim Red	32	London No. 2 Lake Park

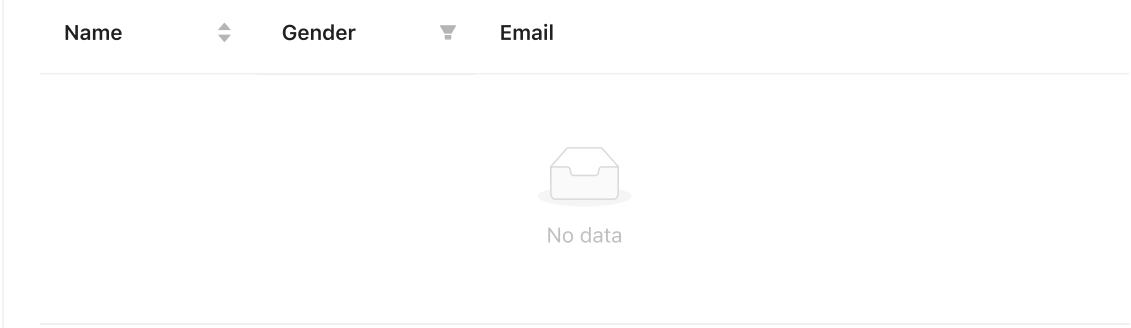
1

Customized filter panel

Implement a customized column search example via `filterDropdown`.

Add the `boolean` type parameter `closeDropdown` to the function `clearFilters`. Whether to close the filter menu is `true` by default. Add the `boolean` type parameter `confirm` to clear whether to submit the option during filtering. The default is `true`.

```
import React, { useRef, useState } from 'react';
import { SearchOutlined } from '@ant-design/icons';
import type { InputRef } from 'antd';
import { Button, Input, Space, Table } from 'antd';
import type { ColumnsType, ColumnType } from 'antd/es/table';
import type { FilterConfirmProps } from 'antd/es/table/interface';
import Highlighter from 'react-highlight-words';

interface DataType {
  key: string;
  name: string;
  age: number;
  address: string;
}

type DataIndex = keyof DataType;

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



Ajax

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 React, { useEffect, useState } from 'react';
import { Table } from 'antd';
import type { ColumnsType, TablePaginationConfig } from 'antd/es/table';
import type { FilterValue, SorterResult } from 'antd/es/table/interface';
import qs from 'qs';

interface DataType {
  name: {
    first: string;
    last: string;
  };
  gender: string;
  email: string;
  login: {
    uuid: string;
  };
}

interface TableParams {
  pagination?: TablePaginationConfig;
  sortField?: string;
  sortOrder?: string;
  filters?: Record<string, FilterValue>;
}

const columns: ColumnsType<DataType> = [
  {
    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: TableParams) => ({
  results: params.pagination?.pageSize,
```

Middle size table

Name	Age	Address
John Brown	32	New York No. 1 Lake Park
Jim Green	42	London No. 1 Lake Park
Joe Black	32	Sydney No. 1 Lake Park

1

Small size table

Name	Age	Address
John Brown	32	New York No. 1 Lake Park
Jim Green	42	London No. 1 Lake Park
Joe Black	32	Sydney No. 1 Lake Park

1

size

There are two compacted table sizes: `middle` and `small`. The `small` size is used in Modals only.

```
import React from 'react';
import { Table, Divider } from 'antd';
import type { ColumnsType } from 'antd/es/table';

interface DataType {
  key: React.Key;
  name: string;
  age: number;
  address: string;
}

const columns: ColumnsType<DataType> = [
  {
    title: 'Name',
    dataIndex: 'name',
  },
  {
    title: 'Age',
    dataIndex: 'age',
  },
  {
    title: 'Address',
    dataIndex: 'address',
  },
];

const data: DataType[] = [
  {
    key: '1',
    name: 'John Brown',
    age: 32,
    address: 'New York No. 1 Lake Park',
  },
  {
    key: '2',
    name: 'Jim Green',
```

Header

Name	Cash Assets	Address
John Brown	￥300,000.00	New York No. 1 Lake Park
Jim Green	￥1,256,000.00	London No. 1 Lake Park
Joe Black	￥120,000.00	Sydney No. 1 Lake Park

Footer

1

border, title and footer

Add border, title and footer for table.

```
import React from 'react';
import { Table } from 'antd';
import type { ColumnsType } from 'antd/es/table';

interface DataType {
  key: string;
  name: string;
  money: string;
  address: string;
}

const columns: ColumnsType<DataType> = [
  {
    title: 'Name',
    dataIndex: 'name',
    render: (text) => <a>{text}</a>,
  },
  {
    title: 'Cash Assets',
    className: 'column-money',
    dataIndex: 'money',
    align: 'right',
  },
  {
    title: 'Address',
    dataIndex: 'address',
  },
];

const data: DataType[] = [
  {
    key: '1',
    name: 'John Brown',
    money: '￥300,000.00',
    address: 'New York No. 1 Lake Park',
  },
  {
    key: '2',
    name: 'Jim Green',
    money: '￥1,256,000.00',
    address: 'London No. 1 Lake Park',
  },
  {
    key: '3',
    name: 'Joe Black',
    money: '￥120,000.00',
```

	Name	Age	Address	Action
☐	John Brown	32	New York No. 1 Lake Park	Delete
☐	Jim Green	42	London No. 1 Lake Park	Delete
	Not Expandable	29	Jiangsu No. 1 Lake Park	Delete
☐	Joe Black	32	Sydney No. 1 Lake Park	Delete

< 1 >

Expandable Row

When there's too much information to show and the table can't display all at once.

```
import React from 'react';
import { Table } from 'antd';
import type { ColumnsType } from 'antd/es/table';

interface DataType {
  key: React.Key;
  name: string;
  age: number;
  address: string;
  description: string;
}

const columns: ColumnsType<DataType> = [
  { title: 'Name', dataIndex: 'name', key: 'name' },
  { title: 'Age', dataIndex: 'age', key: 'age' },
  { title: 'Address', dataIndex: 'address', key: 'address' },
  {
    title: 'Action',
    dataIndex: '',
    key: 'x',
    render: () => <a>Delete</a>,
  },
];

const data: DataType[] = [
  {
    key: 1,
    name: 'John Brown',
    age: 32,
    address: 'New York No. 1 Lake Park',
    description: 'My name is John Brown, I am 32 years old, living in New York No. 1 Lake Park.'
  },
  {
    key: 2,
    name: 'Jim Green',
    age: 42,
    address: 'London No. 1 Lake Park',
    description: 'My name is Jim Green, I am 42 years old, living in London No. 1 Lake Park.',
  },
  {
    key: 3,
    name: 'Not Expandable',
    age: 29,
    address: 'Jiangsu No. 1 Lake Park',
    description: 'This not expandable',
  },
  {
    key: 4,
```

Name	Age	Address
John Brown	32	New York No. 1 Lake Park
Jim Green	42	London No. 1 Lake Park
Not Expandable	29	Jiangsu No. 1 Lake Park
Joe Black	32	Sydney No. 1 Lake Park

Order Specific Column

You can control the order of the expand and select columns by using `Table.EXPAND_COLUMN` and `Table.SELECT_COLUMN`.

```
import React from 'react';
import { Table } from 'antd';
import type { ColumnsType } from 'antd/es/table';

interface DataType {
  key: React.Key;
  name: string;
  age: number;
  address: string;
  description: string;
}

const columns: ColumnsType<DataType> = [
  { title: 'Name', dataIndex: 'name', key: 'name' },
  Table.EXPAND_COLUMN,
  { title: 'Age', dataIndex: 'age', key: 'age' },
  Table.SELECTION_COLUMN,
  { title: 'Address', dataIndex: 'address', key: 'address' },
];

const data: DataType[] = [
  {
    key: 1,
    name: 'John Brown',
    age: 32,
    address: 'New York No. 1 Lake Park',
    description: 'My name is John Brown, I am 32 years old, living in New York No. 1 Lake Park.'
  },
  {
    key: 2,
    name: 'Jim Green',
    age: 42,
    address: 'London No. 1 Lake Park',
    description: 'My name is Jim Green, I am 42 years old, living in London No. 1 Lake Park.',
  },
  {
    key: 3,
    name: 'Not Expandable',
    age: 29,
    address: 'Jiangsu No. 1 Lake Park',
    description: 'This not expandable',
  },
  {
    key: 4,
    name: 'Joe Black',
    age: 32,
    address: 'Sydney No. 1 Lake Park',
```

RowHead	Name	Age	Home phone		Address
1	John Brown	32	0571-22098909	18889898989	New York No. 1 Lake Park
2	Jim Green				
3	Joe Black	32	0575-22098909	18900010002	Sydney No. 1 Lake Park
4	Jim Red	18	0575-22098909	18900010002	London No. 2 Lake Park
5	Jake White	18		18900010002	Dublin No. 2 Lake Park

1

colSpan and rowSpan

Table column title supports `colSpan` that set in `column`.

Table cell supports `colSpan` and `rowSpan` that set in render return object. When each of them is set to `0`, the cell will not be rendered.

```
import React from 'react';
import { Table } from 'antd';
import type { ColumnsType } from 'antd/es/table';

interface DataType {
  key: string;
  name: string;
  age: number;
  tel: string;
  phone: number;
  address: string;
}

// In the fifth row, other columns are merged into first column
// by setting it's colSpan to be 0
const sharedOnCell = (_: DataType, index: number) => {
  if (index === 1) {
    return { colSpan: 0 };
  }

  return {};
};

const columns: ColumnsType<DataType> = [
  {
    title: 'RowHead',
    dataIndex: 'key',
    rowScope: 'row',
  },
  {
    title: 'Name',
    dataIndex: 'name',
    render: (text) => <a>{text}</a>,
    onCell: (_, index) => ({
      colSpan: index === 1 ? 5 : 1,
    }),
  },
  {
    title: 'Age',
    dataIndex: 'age',
```

CheckStrictly:

Name	Age	Address
John Brown sr.	60	New York No. 1 Lake Park
Joe Black	32	Sydney No. 1 Lake Park

1

Tree data

Display tree structure data in Table when there is field key `children` in dataSource, try to customize `childrenColumnName` property to avoid tree table structure.

You can control the indent width by setting `indentSize`.

```
import React, { useState } from 'react';
import { Space, Switch, Table } from 'antd';
import type { ColumnsType } from 'antd/es/table';
import type { TableRowSelection } from 'antd/es/table/interface';

interface DataType {
  key: React.ReactNode;
  name: string;
  age: number;
  address: string;
  children?: DataType[];
}

const columns: ColumnsType<DataType> = [
  {
    title: 'Name',
    dataIndex: 'name',
    key: 'name',
  },
  {
    title: 'Age',
    dataIndex: 'age',
    key: 'age',
    width: '12%',
  },
  {
    title: 'Address',
    dataIndex: 'address',
    width: '30%',
    key: 'address',
  },
];

const data: DataType[] = [
  {
    key: 1,
    name: 'John Brown sr.',
    age: 60,
    address: 'New York No. 1 Lake Park',
    children: [
      {
        key: 11,
        name: 'John Brown',
        age: 42,
        address: 'New York No. 2 Lake Park',
      },
      {
        key: 12,
        name: 'John Brown jr.',
```

Edward King 0	32	London, Park Lane no. 0
Edward King 1	32	London, Park Lane no. 1
Edward King 2	32	London, Park Lane no. 2
Edward King 3	32	London, Park Lane no. 3

1 2 50 / page

Fixed Header

Display large amounts of data in scrollable view.

> Specify width of columns if header and cell do not align properly. If specified width is not working or have gutter between columns, please try to leave one column at least without width to fit fluid layout, or make sure no long word to break table layout.

```
import React from 'react';
import { Table } from 'antd';
import type { ColumnsType } from 'antd/es/table';

interface DataType {
  key: React.Key;
  name: string;
  age: number;
  address: string;
}

const columns: ColumnsType<DataType> = [
  {
    title: 'Name',
    dataIndex: 'name',
    width: 150,
  },
  {
    title: 'Age',
    dataIndex: 'age',
    width: 150,
  },
  {
    title: 'Address',
    dataIndex: 'address',
  },
];

const data: DataType[] = [];
for (let i = 0; i < 100; i++) {
  data.push({
    key: i,
    name: `Edward King ${i}`,
    age: 32,
    address: `London, Park Lane no. ${i}`,
  });
}

const App: React.FC = () => (
  <Table columns={columns} dataSource={data} pagination={{ pageSize: 50 }} scroll={{ y: 240 }} /
);

export default App;
```

Full Name	Age	Column 1	Column 2	Column 3	Colur	Action
John Brown	32	New York Park	New York Park	New York Park	New ` Park	action
Jim Green	40	London Park	London Park	London Park	Lond(	action

1

Fixed Columns

To fix some columns and scroll inside other columns, and you must set `scroll.x` meanwhile.

> Specify the width of columns if header and cell do not align properly. If specified width is not working or have gutter between columns, please try to leave one column at least without width to fit fluid layout, or make sure no long word to break table layout.
> A fixed value which is greater than table width for `scroll.x` is recommended. The sum of unfixed columns should not greater than `scroll.x` .

Note: v4 using sticky to implement fixed effect. IE 11 will downgrade to horizontal scroll.

```
import React from 'react';
import { Table } from 'antd';
import type { ColumnsType } from 'antd/es/table';

interface DataType {
  key: React.Key;
  name: string;
  age: number;
  address: string;
}

const columns: ColumnsType<DataType> = [
  {
    title: 'Full Name',
    width: 100,
    dataIndex: 'name',
    key: 'name',
    fixed: 'left',
  },
  {
    title: 'Age',
    width: 100,
    dataIndex: 'age',
    key: 'age',
    fixed: 'left',
    sorter: true,
  },
  { title: 'Column 1', dataIndex: 'address', key: '1' },
  { title: 'Column 2', dataIndex: 'address', key: '2' },
  { title: 'Column 3', dataIndex: 'address', key: '3' },
  { title: 'Column 4', dataIndex: 'address', key: '4' },
  { title: 'Column 5', dataIndex: 'address', key: '5' },
  { title: 'Column 6', dataIndex: 'address', key: '6' },
  { title: 'Column 7', dataIndex: 'address', key: '7' },
  { title: 'Column 8', dataIndex: 'address', key: '8' },
  {
    title: 'Action',
    key: 'operation',
    fixed: 'right',
    width: 100,
    render: () => <a>action</a>,
  },
];

const data: DataType[] = [
```

Edward 0	32	London Park no. 0	London Park no. 0	London Park no.	action
Edward 1	32	London Park no. 1	London Park no. 1	London Park no.	action
Edward 2	32	London Park no. 2	London Park no. 2	London Park no.	action
Edward 3	32	London Park no. 3	London Park no. 3	London Park no.	action
Edward 4	32	London Park no. 4	London Park no. 4	London Park no.	action

1 2 3 4 5 ••• 10 > 10 / page

Fixed Columns and Header

A Solution for displaying large amounts of data with long columns.

> Specify the width of columns if header and cell do not align properly. If specified width is not working or have gutter between columns, please try to leave one column at least without width to fit fluid layout, or make sure no long word to break table layout.
> A fixed value which is greater than table width for `scroll.x` is recommended. The sum of unfixed columns should not greater than `scroll.x`.

```
import React from 'react';
import { Table } from 'antd';
import type { ColumnsType } from 'antd/es/table';

interface DataType {
  key: React.Key;
  name: string;
  age: number;
  address: string;
}

const columns: ColumnsType<DataType> = [
  {
    title: 'Full Name',
    width: 100,
    dataIndex: 'name',
    key: 'name',
    fixed: 'left',
  },
  {
    title: 'Age',
    width: 100,
    dataIndex: 'age',
    key: 'age',
    fixed: 'left',
  },
  {
    title: 'Column 1',
    dataIndex: 'address',
    key: '1',
    width: 150,
  },
  {
    title: 'Column 2',
    dataIndex: 'address',
    key: '2',
```

John Brown	1	Lake Park	C	2035	Lake S	M
John Brown	2	Lake Park	C	2035	Lake S	M
John Brown	3	Lake Park	C	2035	Lake S	M
John Brown	4	Lake Park	C	2035	Lake S	M
John Brown	5	Lake Park	C	2035	Lake S	M

1 2 3 4 5 ••• 10 > 10 / page

Grouping table head

Group table head with `columns[n].children`.

```
import React from 'react';
import { Table } from 'antd';
import type { ColumnsType } from 'antd/es/table';

interface DataType {
  key: React.Key;
  name: string;
  age: number;
  street: string;
  building: string;
  number: number;
  companyAddress: string;
  companyName: string;
  gender: string;
}

const columns: ColumnsType<DataType> = [
  {
    title: 'Name',
    dataIndex: 'name',
    key: 'name',
    width: 100,
    fixed: 'left',
    filters: [
      {
        text: 'Joe',
        value: 'Joe',
      },
      {
        text: 'John',
        value: 'John',
      },
    ],
    onFilter: (value: string, record) => record.name.indexOf(value) === 0,
  },
  {
    title: 'Other',
    children: [
      {
        title: 'Age',
        dataIndex: 'age',
```

Add a row

name	age	address	operation
Edward King 0	32	London, Park Lane no. 0	Delete
Edward King 1	32	London, Park Lane no. 1	Delete

1

Editable Cells

Table with editable cells. When work with `shouldCellUpdate`, please take care of closure.

```
import React, { useContext, useEffect, useRef, useState } from 'react';
import type { InputRef } from 'antd';
import { Button, Form, Input, Popconfirm, Table } from 'antd';
import type { FormInstance } from 'antd/es/form';

const EditableContext = React.createContext<FormInstance<any> | null>(null);

interface Item {
  key: string;
  name: string;
  age: string;
  address: string;
}

interface EditableRowProps {
  index: number;
}

const EditableRow: React.FC<EditableRowProps> = ({ index, ...props }) => {
  const [form] = Form.useForm();
  return (
    <Form form={form} component={false}>
      <EditableContext.Provider value={form}>
        <tr {...props} />
      </EditableContext.Provider>
    </Form>
  );
};

interface EditableCellProps {
  title: React.ReactNode;
  editable: boolean;
  children: React.ReactNode;
  dataIndex: keyof Item;
  record: Item;
  handleSave: (record: Item) => void;
}

const EditableCell: React.FC<EditableCellProps> = ({
  title,
  editable,
  children,
  dataIndex,
  record,
  handleSave,
  ...restProps
}) => {
  const [editing, setEditing] = useState(false);
  const inputRef = useRef<InputRef>(null);
  const form = useContext(EditableContext)!;
```

name	age	address	operation
Edward 0	32	London Park no. 0	Edit
Edward 1	32	London Park no. 1	Edit
Edward 2	32	London Park no. 2	Edit
Edward 3	32	London Park no. 3	Edit
Edward 4	32	London Park no. 4	Edit
Edward 5	32	London Park no. 5	Edit
Edward 6	32	London Park no. 6	Edit
Edward 7	32	London Park no. 7	Edit
Edward 8	32	London Park no. 8	Edit
Edward 9	32	London Park no. 9	Edit

< 1 2 3 4 5 ••• 10 > 10 / page

Editable Rows

Table with editable rows.

```
import React, { useState } from 'react';
import { Form, Input, InputNumber, Popconfirm, Table, Typography } from 'antd';

interface Item {
  key: string;
  name: string;
  age: number;
  address: string;
}

const originData: Item[] = [];
for (let i = 0; i < 100; i++) {
  originData.push({
    key: i.toString(),
    name: `Edward ${i}`,
    age: 32,
    address: `London Park no. ${i}`,
  });
}
interface EditableCellProps extends React.HTMLAttributes<HTMLElement> {
  editing: boolean;
  dataIndex: string;
  title: any;
  inputType: 'number' | 'text';
  record: Item;
  index: number;
  children: React.ReactNode;
}

const EditableCell: React.FC<EditableCellProps> = ({
  editing,
```

Name	Platform	Version	Upgraded	Creator	Date	Action
Screen	iOS	10.3.4.5654	500	Jack	2014-12-24 23:12:00	Publish

Date	Name	Status	Upgrade Status	Action
2014-12-24 23:12:00	This is production name	Finished	Upgraded: 56	Pause Stop More
2014-12-24 23:12:00	This is production name	Finished	Upgraded: 56	Pause Stop More
2014-12-24 23:12:00	This is production name	Finished	Upgraded: 56	Pause Stop More

Name	Platform	Version	Upgraded	Creator	Date	Action
Screen	iOS	10.3.4.5654	500	Jack	2014-12-24 23:12:00	Publish
Screen	iOS	10.3.4.5654	500	Jack	2014-12-24 23:12:00	Publish

< 1 >

Name	Platform	Version	Upgraded	Creator	Date	Action
Screen	iOS	10.3.4.5654	500	Jack	2014-12-24 23:12:00	Publish

Date	Name	Status	Upgrade Status	Action
2014-12-24 23:12:00	This is production name	Finished	Upgraded: 56	Pause Stop More
2014-12-24 23:12:00	This is production name	Finished	Upgraded: 56	Pause Stop More
2014-12-24 23:12:00	This is production name	Finished	Upgraded: 56	Pause Stop More

Name	Platform	Version	Upgraded	Creator	Date	Action
Screen	iOS	10.3.4.5654	500	Jack	2014-12-24 23:12:00	Publish
Screen	iOS	10.3.4.5654	500	Jack	2014-12-24 23:12:00	Publish

< 1 >

Name	Platform	Version	Upgraded	Creator	Date	Action
Screen	iOS	10.3.4.5654	500	Jack	2014-12-24 23:12:00	Publish

Date	Name	Status	Upgrade Status	Action
2014-12-24 23:12:00	This is production name	Finished	Upgraded: 56	Pause Stop More
2014-12-24	This is production			More

Name	Age	Address
John Brown	32	Long text Long text
Jim Green	42	London No. 1 Lake Park
Joe Black	32	Sidney No. 1 Lake Park

Drag sorting

By using `components`, we can integrate table with dnd-kit to implement drag sorting function.

```
import type { DragEndEvent } from '@dnd-kit/core';
import { DndContext } from '@dnd-kit/core';
import {
  arrayMove,
  SortableContext,
  useSortable,
  verticalListSortingStrategy,
} from '@dnd-kit/sortable';
import { CSS } from '@dnd-kit/utilities';
import { Table } from 'antd';
import type { ColumnsType } from 'antd/es/table';
import React, { useState } from 'react';

interface DataType {
  key: string;
  name: string;
  age: number;
  address: string;
}

const columns: ColumnsType<DataType> = [
  {
    title: 'Name',
    dataIndex: 'name',
  },
  {
    title: 'Age',
    dataIndex: 'age',
  },
  {
    title: 'Address',
    dataIndex: 'address',
  },
];

interface RowProps extends React.HTMLAttributes<HTMLTableRowElement> {
  'data-row-key': string;
}

const Row = (props: RowProps) => {
  const { attributes, listeners, setNodeRef, transform, transition, isDragging } = useSortable({
    id: props['data-row-key'],
  });

  const style: React.CSSProperties = {
    ...props.style,
```

	Name	Age	Address
☰	John Brown	32	Long text Long text
☰	Jim Green	42	London No. 1 Lake Park
☰	Joe Black	32	Sidney No. 1 Lake Park

Drag sorting with handler

Alternatively you can implement drag sorting with handler using dnd-kit.

```
import { MenuOutlined } from '@ant-design/icons';
import type { DragEndEvent } from '@dnd-kit/core';
import { DndContext } from '@dnd-kit/core';
import {
  arrayMove,
  SortableContext,
  useSortable,
  verticalListSortingStrategy,
} from '@dnd-kit/sortable';
import { CSS } from '@dnd-kit/utilities';
import { Table } from 'antd';
import type { ColumnsType } from 'antd/es/table';
import React, { useState } from 'react';

interface DataType {
  key: string;
  name: string;
  age: number;
  address: string;
}

const columns: ColumnsType<DataType> = [
  {
    key: 'sort',
  },
  {
    title: 'Name',
    dataIndex: 'name',
  },
  {
    title: 'Age',
    dataIndex: 'age',
  },
  {
    title: 'Address',
    dataIndex: 'address',
  },
];

interface RowProps extends React.HTMLAttributes<HTMLTableRowElement> {
  'data-row-key': string;
}

const Row = ({ children, ...props }: RowProps) => {
  const {
    attributes
```

Name	Age	Address	Long Colu...	Long Colu...	Long Column
John Brown	32	New York N...	New York N...	New York N...	New York N...
Jim Green	42	London No. ...	London No. ...	London No. ...	London No. ...
Joe Black	32	Sydney No. ...	Sydney No. ...	Sydney No. ...	Sydney No. ...

1

ellipsis column

Ellipsis cell content via setting `column.ellipsis`.

> Cannot ellipsis table header with sorters and filters for now.

```
import React from 'react';
import { Table } from 'antd';
import type { ColumnsType } from 'antd/es/table';

interface DataType {
  key: React.Key;
  name: string;
  age: number;
  address: string;
}

const columns: ColumnsType<DataType> = [
  {
    title: 'Name',
    dataIndex: 'name',
    key: 'name',
    render: (text) => <a>{text}</a>,
    width: 150,
  },
  {
    title: 'Age',
    dataIndex: 'age',
    key: 'age',
    width: 80,
  },
  {
    title: 'Address',
    dataIndex: 'address',
    key: 'address 1',
    ellipsis: true,
  },
  {
    title: 'Long Column Long Column Long Column',
    dataIndex: 'address',
    key: 'address 2',
    ellipsis: true,
  },
  {
    title: 'Long Column Long Column',
    dataIndex: 'address',
    key: 'address 3',
    ellipsis: true,
  },
  {
    title: 'Long Column',
    dataIndex: 'address',
    key: 'address 4',
    ellipsis: true,
  },
```

Name	Age	Address	Long Colu...	Long Colu...	Long Column
John Brown	32	New York N...	New York N...	New York N...	New York N...
Jim Green	42	London No. ...	London No. ...	London No. ...	London No. ...
Joe Black	32	Sydney No. ...	Sydney No. ...	Sydney No. ...	Sydney No. ...

ellipsis column custom tooltip

Ellipsis cell content via setting `column.ellipsis.showTitle`, use `Tooltip` instead of the html title attribute.

```
import React from 'react';
import { Table, Tooltip } from 'antd';
import type { ColumnsType } from 'antd/es/table';

interface DataType {
  key: React.Key;
  name: string;
  age: number;
  address: string;
}

const columns: ColumnsType<DataType> = [
  {
    title: 'Name',
    dataIndex: 'name',
    key: 'name',
    render: (text) => <a>{text}</a>,
    width: 150,
  },
  {
    title: 'Age',
    dataIndex: 'age',
    key: 'age',
    width: 80,
  },
  {
    title: 'Address',
    dataIndex: 'address',
    key: 'address 1',
    ellipsis: {
      showTitle: false,
    },
    render: (address) => (
      <Tooltip placement="topLeft" title={address}>
        {address}
      </Tooltip>
    ),
  },
  {
    title: 'Long Column Long Column Long Column',
    dataIndex: 'address',
    key: 'address 2',
    ellipsis: {
      showTitle: false,
    },
    render: (address) => (
      <Tooltip placement="topLeft" title={address}>
        {address}
      </Tooltip>
    ),
  },
```

Name	Borrow	Repayment
John Brown	10	33
Jim Green	100	0
Joe Black	10	10
Jim Red	75	45
Total	195	88
Balance	107	

Light	Everything that has a beginning, has an end.
Bamboo	Everything that has a beginning, has an end.
Little	Everything that has a beginning, has an end.
Light	Everything that has a beginning, has an end.
Bamboo	Everything that has a beginning, has an end.
Little	Everything that has a beginning, has an end.
Light	Everything that has a beginning, has an end.
Bamboo	Everything that has a beginning, has an end.
Little	Everything that has a beginning, has an end.

Summary

Set summary content by `summary` prop. Sync column fixed status with `Table.Summary.Cell`. You can fixed it by set `Table.Summary` `fixed` prop(since `4.16.0`).

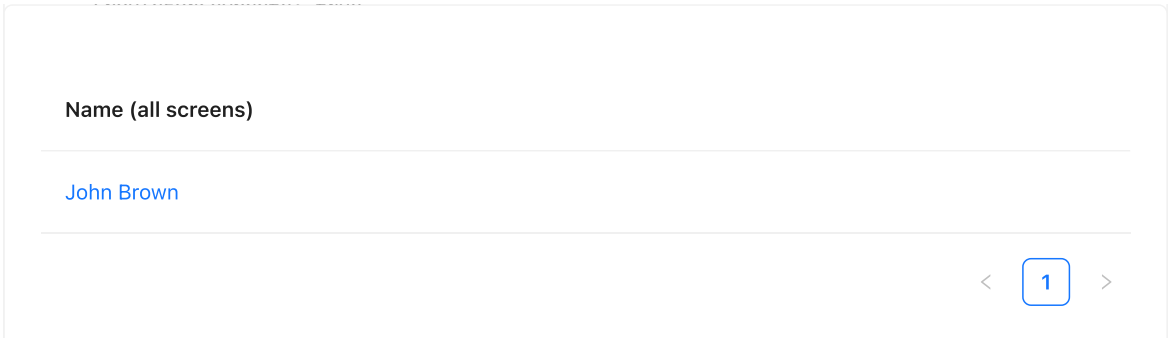
```
import React from 'react';
import { Table, Typography } from 'antd';
import type { ColumnsType } from 'antd/es/table';

const { Text } = Typography;

interface DataType {
  key: string;
  name: string;
  borrow: number;
  repayment: number;
}
```

A	E	F

Virtual list

Integrate virtual scroll with `react-window` to achieve a high performance table of 100,000 data.

```
import type { TableProps } from 'antd';
import { Table, theme } from 'antd';
import classNames from 'classnames';
import ResizeObserver from 'rc-resize-observer';
import React, { useEffect, useRef, useState } from 'react';
import { VariableSizeGrid as Grid } from 'react-window';

const VirtualTable = <RecordType extends object>(props: TableProps<RecordType>) => {
  const { columns, scroll } = props;
  const [tableWidth, setTableWidth] = useState(0);
  const { token } = theme.useToken();

  const widthColumnCount = columns!.filter(({ width }) => !width).length;
  const mergedColumns = columns!.map((column) => {
    if (column.width) {
      return column;
    }

    return {
      ...column,
      width: Math.floor(tableWidth / widthColumnCount),
    };
  });

  const gridRef = useRef<any>();
  const [connectObject] = useState<any>(() => {
    const obj = {};
    Object.defineProperty(obj, 'scrollLeft', {
      get: () => {
        if (gridRef.current) {
          return gridRef.current?.state?.scrollLeft;
        }
        return null;
      },
      set: (scrollLeft: number) => {
        if (gridRef.current) {
          gridRef.current.scrollTo({ scrollLeft });
        }
      },
    });

    return obj;
  });

  const resetVirtualGrid = () => {
    gridRef.current?.resetAfterIndices({
      columnIndex: 0,
      shouldForceUpdate: true
```



Responsive

Responsive columns.

```
import React from 'react';
import { Table } from 'antd';
import type { ColumnsType } from 'antd/es/table';

interface DataType {
  key: React.Key;
  name: string;
  age: number;
  address: string;
}

const columns: ColumnsType<DataType> = [
  {
    title: 'Name (all screens)',
    dataIndex: 'name',
    key: 'name',
    render: (text) => <a>{text}</a>,
  },
  {
    title: 'Age (medium screen or bigger)',
    dataIndex: 'age',
    key: 'age',
    responsive: ['md'],
  },
  {
    title: 'Address (large screen or bigger)',
    dataIndex: 'address',
    key: 'address',
    responsive: ['lg'],
  },
];

const data: DataType[] = [
  {
    key: '1',
    name: 'John Brown',
    age: 32,
    address: 'New York No. 1 Lake Park',
  },
];

const App: React.FC = () => <Table columns={columns} dataSource={data} />;

export default App;
```

```
    >
      <Table
        {...props}
        className="virtual-table"
        columns={mergedColumns}
        pagination={false}
        components={{
          body: renderVirtualList,
        }}
      />
    </ResizeObserver>
  );
};
```

topLeft topCenter topRight none

bottomLeft bottomCenter bottomRight none

1

Name	Age	Address	Tags	Action
John Brown	32	New York No. 1 Lake Park	NICE DEVELOPER	Invite John Brown Delete
Jim Green	42	London No. 1 Lake Park	LOSER	Invite Jim Green Delete
Joe Black	32	Sydney No. 1 Lake Park	COOL TEACHER	Invite Joe Black Delete

1

Pagination Settings

Table pagination settings.

```
import React, { useState } from 'react';
import { Radio, Space, Table, Tag } from 'antd';
import type { ColumnsType } from 'antd/es/table';

interface DataType {
  key: string;
  name: string;
  age: number;
  address: string;
  tags: string[];
}

type TablePaginationPosition =
  | 'topLeft'
  | 'topCenter'
  | 'topRight'
  | 'bottomLeft'
  | 'bottomCenter'
  | 'bottomRight';

const topOptions = [
  { label: 'topLeft', value: 'topLeft' },
  { label: 'topCenter', value: 'topCenter' },
  { label: 'topRight', value: 'topRight' },
  { label: 'none', value: 'none' },
];

const bottomOptions = [
  { label: 'bottomLeft', value: 'bottomLeft' },
  { label: 'bottomCenter', value: 'bottomCenter' },
  { label: 'bottomRight', value: 'bottomRight' },
  { label: 'none', value: 'none' },
];

const columns: ColumnsType<DataType> = [
  {
    title: 'Name',
    dataIndex: 'name',
    key: 'name',
    render: (text) => <a>{text}</a>,
  },
  {
    title: 'Age',
```

Edward 0	32	London Park no. 0	London Park no. 0	London Park no.	action
Edward 1	32	London Park no. 1	London Park no. 1	London Park no.	action
Edward 2	32	London Park no. 2	London Park no. 2	London Park no.	action
Edward 3	32	London Park no. 3	London Park no. 3	London Park no.	action
Edward 4	32	London Park no. 4	London Park no. 4	London Park no.	action
Edward 5	32	London Park no. 5	London Park no. 5	London Park no.	action
Edward 6	32	London Park no. 6	London Park no. 6	London Park no.	action
Edward 7	32	London Park no. 7	London Park no. 7	London Park no.	action
Edward 8	32	London Park no. 8	London Park no. 8	London Park no.	action
Edward 9	32	London Park no. 9	London Park no. 9	London Park no.	action

< 1 2 3 4 5 ••• 10 > 10 / page

Fixed header and scroll bar with the page

For long table，need to scroll to view the header and scroll bar，then you can now set the fixed header and scroll bar to follow the page.

```
import React, { useState } from 'react';
import { Switch, Table } from 'antd';
import type { ColumnsType } from 'antd/es/table';

interface DataType {
  key: React.Key;
  name: string;
  age: number;
  address: string;
}

const columns: ColumnsType<DataType> = [
  {
    title: 'Full Name',
    width: 100,
    dataIndex: 'name',
    key: 'name',
    fixed: 'left',
  },
  {
    title: 'Age',
    width: 100,
    dataIndex: 'age',
    key: 'age',
```

Bordered: loading: Title: Column Header: Footer:

Expandable: Checkbox: Fixed Header: Has Data: Ellipsis:

Size: Large Middle Small Table Scroll: Unset Scroll Fixed Columns

Table Layout: Unset Fixed Pagination Top: TopLeft TopCenter TopRight None

Pagination Bottom: BottomLeft BottomCenter BottomRight None

	Name	Age	Address	Action
	John Brown	12	New York No. 1 Lake Park	Delete More actions
	John Brown	22	New York No. 2 Lake Park	Delete More actions
	John Brown	32	New York No. 3 Lake Park	Delete More actions
	John Brown	42	New York No. 4 Lake Park	Delete More actions
	John Brown	52	New York No. 5 Lake Park	Delete More actions
	John Brown	62	New York No. 6 Lake Park	Delete More actions
	John Brown	72	New York No. 7 Lake Park	Delete More actions
	John Brown	82	New York No. 8 Lake Park	Delete More actions
	John Brown	92	New York No. 9 Lake Park	Delete More actions
	John Brown	102	New York No. 10 Lake Park	Delete More actions

Here is footer

1

Dynamic Settings

Select different settings to see the result.

```
import React, { useState } from 'react';
import { DownOutlined } from '@ant-design/icons';
import type { RadioChangeEvent } from 'antd';
import { Form, Radio, Space, Switch, Table } from 'antd';
import type { SizeType } from 'antd/es/config-provider/SizeContext';
import type { ColumnsType, TableProps } from 'antd/es/table';
import type { ExpandableConfig, TableRowSelection } from 'antd/es/table/interface';

interface DataType {
  key: number;
  name: string;
  age: number;
  address: string;
  description: string;
}

type TablePaginationPosition =
  | 'topLeft'
```

```
  | 'topLeft'
  | 'topCenter'
  | 'topRight'
  | 'bottomLeft'
  | 'bottomCenter'
  | 'bottomRight';

const columns: ColumnsType<DataType> = [
  {
    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 as string) === 0,
  },
  {
    title: 'Action',
    key: 'action',
    sorter: true,
    render: () => (
      <Space size="middle">
        <a>Delete</a>
        <a>
          <Space>
            More actions
            <DownOutlined />
          </Space>
        </a>
      </Space>
    ),
  },
];

const data: DataType[] = [];
for (let i = 1; i <= 10; i++) {
  data.push({
    key: i,
    name: 'John Brown',
    age: Number(`${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
  });
}

const defaultExpandable = { expandedRowRender: (record: DataType) => <p>{record.description}</p>
const defaultTitle = () => 'Here is title';
const defaultFooter = () => 'Here is footer';

const App: React.FC = () => {
  const [bordered, setBordered] = useState(false);
  const [loading, setLoading] = useState(false);
  const [size, setSize] = useState<SizeType>('large');
  const [expandable, setExpandable] = useState<ExpandableConfig<DataType> | undefined>(
    defaultExpandable,
  );
  const [showTitle, setShowTitle] = useState(false);
  const [showHeader, setShowHeader] = useState(true);
  const [showFooter, setShowFooter] = useState(true);
```

```
const [rowSelection, setRowSelection] = useState<TableRowSelection<DataType> | undefined>({});
const [hasData, setHasData] = useState(true);
const [tableLayout, setTableLayout] = useState();
const [top, setTop] = useState<TablePaginationPosition | 'none'>('none');
const [bottom, setBottom] = useState<TablePaginationPosition>('bottomRight');
const [ellipsis, setEllipsis] = useState(false);
const [yScroll, setYScroll] = useState(false);
const [xScroll, setXScroll] = useState<string>();

const handleBorderChange = (enable: boolean) => {
  setBordered(enable);
};

const handleLoadingChange = (enable: boolean) => {
  setLoading(enable);
};

const handleSizeChange = (e: RadioChangeEvent) => {
  setSize(e.target.value);
};

const handleTableLayoutChange = (e: RadioChangeEvent) => {
  setTableLayout(e.target.value);
};

const handleExpandChange = (enable: boolean) => {
  setExpandable(enable ? defaultExpandable : undefined);
};

const handleEllipsisChange = (enable: boolean) => {
  setEllipsis(enable);
};

const handleTitleChange = (enable: boolean) => {
  setShowTitle(enable);
};

const handleHeaderChange = (enable: boolean) => {
  setShowHeader(enable);
};

const handleFooterChange = (enable: boolean) => {
  setShowFooter(enable);
};

const handleRowSelectionChange = (enable: boolean) => {
  setRowSelection(enable ? {} : undefined);
};

const handleYScrollChange = (enable: boolean) => {
  setYScroll(enable);
};

const handleXScrollChange = (e: RadioChangeEvent) => {
  setXScroll(e.target.value);
};

const handleDataChange = (newHasData: boolean) => {
  setHasData(newHasData);
};

const scroll: { x?: number | string; y?: number | string } = {};
if (yScroll) {
  scroll.y = 240;
}
if (xScroll) {
  scroll.x = '100vw';
}

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

```
const tableProps: TableProps<DataType> = {
  bordered,
  loading,
  size,
  expandable,
  title: showTitle ? defaultTitle : undefined,
  showHeader,
  footer: showfooter ? defaultFooter : undefined,
  rowSelection,
  scroll,
  tableLayout,
};

return (
  <>
    <Form
      layout="inline"
      className="components-table-demo-control-bar"
      style={{ marginBottom: 16 }}
    >
      <Form.Item label="Bordered">
        <Switch checked={bordered} onChange={handleBorderChange} />
      </Form.Item>
      <Form.Item label="loading">
        <Switch checked={loading} onChange={handleLoadingChange} />
      </Form.Item>
      <Form.Item label="Title">
        <Switch checked={showTitle} onChange={handleTitleChange} />
      </Form.Item>
      <Form.Item label="Column Header">
        <Switch checked={showHeader} onChange={handleHeaderChange} />
      </Form.Item>
      <Form.Item label="Footer">
        <Switch checked={showfooter} onChange={handleFooterChange} />
      </Form.Item>
      <Form.Item label="Expandable">
        <Switch checked={!!expandable} onChange={handleExpandChange} />
      </Form.Item>
      <Form.Item label="Checkbox">
        <Switch checked={!!rowSelection} onChange={handleRowSelectionChange} />
      </Form.Item>
      <Form.Item label="Fixed Header">
        <Switch checked={!!yScroll} onChange={handleYScrollChange} />
      </Form.Item>
      <Form.Item label="Has Data">
        <Switch checked={!!hasData} onChange={handleDataChange} />
      </Form.Item>
      <Form.Item label="Ellipsis">
        <Switch checked={!!ellipsis} onChange={handleEllipsisChange} />
      </Form.Item>
      <Form.Item label="Size">
        <Radio.Group value={size} onChange={handleSizeChange}>
          <Radio.Button value="large">Large</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={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={tableLayout} onChange={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={top}
```

```
            onChange={(e) => {
              setTop(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={bottom}
            onChange={(e) => {
              setBottom(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
        {...tableProps}
        pagination={{ position: [top as TablePaginationPosition, bottom] }}
        columns={tableColumns}
        dataSource={hasData ? data : []}
        scroll={scroll}
      />
    </>
  );
};

export default App;
```

API

Table

Property	Description	Type	Default
bordered	Whether to show all table borders	boolean	false
columns	Columns of table	ColumnsType[]	-
components	Override default table elements	TableComponents	-
dataSource	Data record array to be displayed	object[]	-
expandable	Config expandable content	expandable	-
footer	Table footer renderer	function(currentPageData)	-
getPopupContainer	The render container of dropdowns in table	(triggerNode) => HTMLElement	() => TableHtmlEleme

Property	Description	Type	Default
loading	Loading status of table	boolean | Spin Props	false
locale	The i18n text including filter, sort, empty text, etc	object	Default Value
pagination	Config of pagination. You can ref table pagination config or full `pagination` document, hide it by setting it to `false`	object | `false`	-
rowClassName	Row's className	function(record, index): string	-
rowKey	Row's unique key, could be a string or function that returns a string	string | function(record): string	`key`
rowSelection	Row selection config	object	-
scroll	Whether the table can be scrollable, config	object	-
showHeader	Whether to show table header	boolean	true
showSorterTooltip	The header show next sorter direction tooltip. It will be set as the property of Tooltip if its type is object	boolean | Tooltip props	true
size	Size of table	`large` | `middle` | `small`	`large`
sortDirections	Supported sort way, could be `ascend`, `descend`	Array	[`ascend`, `descend`]
sticky	Set sticky header and scroll bar	boolean | `{offsetHeader?: number, offsetScroll?: number, getContainer?: () => HTMLElement}`	-
summary	Summary content	(currentData) => ReactNode	-
tableLayout	The table-layout attribute of table element	- | `auto` | `fixed`	- `fixed` when header/columns are fixed, or

Property	Description	Type	Default
			using `column.ellipsis`
title	Table title renderer	function(currentPageData)	-
onChange	Callback executed when pagination, filters or sorter is changed	function(pagination, filters, sorter, extra: { currentDataSource: [], action: `paginate` | `sort` | `filter` })	-
onHeaderRow	Set props on per header row	function(columns, index)	-
onRow	Set props on per row	function(record, index)	-

onRow usage

Same as `onRow` `onHeaderRow` `onCell` `onHeaderCell`

```
<Table
  onRow={(record, rowIndex) => {
    return {
      onClick: (event) => {}, // click row
      onDoubleClick: (event) => {}, // double click row
      onContextMenu: (event) => {}, // right button click row
      onMouseEnter: (event) => {}, // mouse enter row
      onMouseLeave: (event) => {}, // mouse leave row
    };
  }}
  onHeaderRow={(columns, index) => {
    return {
      onClick: () => {}, // click header row
    };
  }}
/>
```

Column

One of the Table `columns` prop for describing the table's columns, Column has the same API.

Property	Description	Type	Def
align	The specify which way that column is aligned	`left` | `right` | `center`	`le`
className	The className of this column	string	-
colSpan	Span of this column's title	number	-
dataIndex	Display field of the data record,	string | string[]	-

Property	Description	Type	De
	support nest path by string array		
defaultFilteredValue	Default filtered values	string[]	-
filterResetToDefaultFilteredValue	click the reset button, whether to restore the default filter	boolean	fa
defaultSortOrder	Default order of sorted values	`ascend` | `descend`	-
ellipsis	The ellipsis cell content, not working with sorter and filters for now. tableLayout would be `fixed` when `ellipsis` is `true` or `{ showTitle?: boolean }`	boolean | {showTitle?: boolean }	fa
filterDropdown	Customized filter overlay	ReactNode | (props: FilterDropdownProps) => ReactNode	-
filterDropdownOpen	Whether `filterDropdown` is visible	boolean	-
filtered	Whether the `dataSource` is filtered	boolean	fa
filteredValue	Controlled filtered value, filter icon will highlight	string[]	-
filterIcon	Customized filter icon	ReactNode | (filtered: boolean) => ReactNode	-
filterMultiple	Whether multiple filters can be selected	boolean	tru
filterMode	To specify the filter interface	'menu' | 'tree'	'me
filterSearch	Whether to be searchable for filter menu	boolean | function(input, record):boolean	fa
filters	Filter menu config	object[]	-
fixed	(IE not support) Set column to be fixed: `true` (same as left) `'left'` `'right'`	boolean | string	fa

Property	Description	Type	De
key	Unique key of this column, you can ignore this prop if you've set a unique `dataIndex`	string	–
render	Renderer of the table cell. The return value should be a ReactNode	function(text, record, index) {}	–
responsive	The list of breakpoints at which to display this column. Always visible if not set	Breakpoint[]	–
rowScope	Set scope attribute for all cells in this column	`row` | `rowgroup`	–
shouldCellUpdate	Control cell render logic	(record, prevRecord) => boolean	–
showSorterTooltip	If header show next sorter direction tooltip, override `showSorterTooltip` in table	boolean | Tooltip props	tru
sortDirections	Supported sort way, override `sortDirections` in `Table`, could be `ascend`, `descend`	Array	[a de
sorter	Sort function for local sort, see Array.sort's compareFunction. If you need sort buttons only, set to `true`	function | boolean	–
sortOrder	Order of sorted values: `ascend` `descend` `null`	`ascend` | `descend` | null	–
title	Title of this column	ReactNode | ({ sortOrder, sortColumn, filters }) => ReactNode	–
width	Width of this column (width not working?)	string | number	–
onCell	Set props on per cell	function(record, rowIndex)	–
onFilter	Function that determines if the row is displayed when filtered	function(value, record) => boolean	–

Property	Description	Type	De
onFilterDropdownOpenChange	Callback executed when `filterDropdownOpen` is changed	function(visible) {}	-
onHeaderCell	Set props on per header cell	function(column)	-

ColumnGroup

Property	Description	Type	Default
title	Title of the column group	ReactNode	-

pagination

Properties for pagination.

Property	Description	Type	Default
position	Specify the position of `Pagination`, could be `topLeft` | `topCenter` | `topRight` | `bottomLeft` | `bottomCenter` | `bottomRight`	Array	[`bottomRight`]

More about pagination, please check `Pagination`.

expandable

Properties for expandable.

Property	Description	Type	Default
childrenColumnName	The column contains children to display	string	children
columnTitle	Set the title of the expand column	ReactNode	-
columnWidth	Set the width of the expand column	string | number	-
defaultExpandAllRows	Expand all rows initially	boolean	false
defaultExpandedRowKeys	Initial expanded row keys	string[]	-
expandedRowClassName	Expanded row's className	function(record, index, indent): string	-
expandedRowKeys	Current expanded row keys	string[]	-
expandedRowRender	Expanded container render for each row	function(record, index, indent,	-

Property	Description	Type	Default
		expanded): ReactNode	
expandIcon	Customize row expand Icon. Ref example	function(props): ReactNode	-
expandRowByClick	Whether to expand row by clicking anywhere in the whole row	boolean	false
fixed	Whether the expansion icon is fixed. Optional true `left` `right`	boolean | string	false
indentSize	Indent size in pixels of tree data	number	15
rowExpandable	Enable row can be expandable	(record) => boolean	-
showExpandColumn	Show expand column	boolean	true
onExpand	Callback executed when the row expand icon is clicked	function(expanded, record)	-
onExpandedRowsChange	Callback executed when the expanded rows change	function(expandedRows)	-

rowSelection

Properties for row selection.

Property	Description	Type	Default
checkStrictly	Check table row precisely; parent row and children rows are not associated	boolean	true
columnTitle	Set the title of the selection column	ReactNode	-
columnWidth	Set the width of the selection column	string | number	`32px`
fixed	Fixed selection column on the left	boolean	-
getCheckboxProps	Get Checkbox or Radio props	function(record)	-
hideSelectAll	Hide the selectAll checkbox and custom selection	boolean	false

Property	Description	Type	Default
preserveSelectedRowKeys	Keep selection `key` even when it removed from `dataSource`	boolean	-
renderCell	Renderer of the table cell. Same as `render` in column	function(checked, record, index, originNode) {}	-
selectedRowKeys	Controlled selected row keys	string[] | number[]	[]
selections	Custom selection config, only displays default selections when set to `true`	object[] | boolean	-
type	`checkbox` or `radio`	`checkbox` | `radio`	`checkbox`
onChange	Callback executed when selected rows change	function(selectedRowKeys, selectedRows, info: { type })	-
onSelect	Callback executed when select/deselect one row	function(record, selected, selectedRows, nativeEvent)	-
onSelectAll	Callback executed when select/deselect all rows	function(selected, selectedRows, changeRows)	-
onSelectInvert	Callback executed when row selection is inverted	function(selectedRowKeys)	-
onSelectNone	Callback executed when row selection is cleared	function()	-
onSelectMultiple	Callback executed when row selection is changed by pressing shift	function(selected, selectedRows, changeRows)	-

scroll

Property	Description	Type	Default
scrollToFirstRowOnChange	Whether to scroll to the top of the table when paging, sorting, filtering changes	boolean	-
x	Set horizontal scrolling, can also be used to specify the width of the scroll area, could be number, percent value, true and 'max-content'	string | number | true	-

Property	Description	Type	Default
y	Set vertical scrolling, can also be used to specify the height of the scroll area, could be string or number	string | number	-

selection

Property	Description	Type	Default
key	Unique key of this selection	string	-
text	Display text of this selection	ReactNode	-
onSelect	Callback executed when this selection is clicked	function(changeableRowKeys)	-

Using in TypeScript

```tsx
import { Table } from 'antd';
import type { ColumnsType } from 'antd/es/table';
import React from 'react';

interface User {
  key: number;
  name: string;
}

const columns: ColumnsType<User> = [
  {
    key: 'name',
    title: 'Name',
    dataIndex: 'name',
  },
];

const data: User[] = [
  {
    key: 0,
    name: 'Jack',
  },
];

const Demo: React.FC = () => (
  <>
    <Table<User> columns={columns} dataSource={data} />
    {/* JSX style usage */}
    <Table<User> dataSource={data}>
      <Table.Column<User> key="name" title="Name" dataIndex="name" />
    </Table>
  </>
);
```

```
export default Demo;
```

Here is the CodeSandbox for TypeScript.

Design Token

▼ Global Token

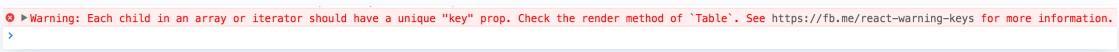
Token Name	Description	Type	Default Value
colorBgContainer	Container background color, e.g: default button, input box, etc. Be sure not to confuse this with `colorBgElevated`.	string	#ffffff
colorBorderSecondary	Slightly lighter than the default border color, this color is the same as `colorSplit`. Solid color is used.	string	#f0f0f0
colorFillAlter	Control the alternative background color of element.	string	rgba(0, 0, 0, 0.02)
colorFillContent	Control the background color of content area.	string	rgba(0, 0, 0, 0.06)
colorFillSecondary	The second level of fill color can outline the shape of the element more clearly, such as Rate, Skeleton, etc. It can also be used as the Hover state of the third level of fill color, such as Table, etc.	string	rgba(0, 0, 0, 0.06)
colorIcon	Weak action. Such as `allowClear` or Alert close button	string	rgba(0, 0, 0, 0.45)
colorIconHover	Weak action hover color. Such as `allowClear` or Alert close button	string	rgba(0, 0, 0, 0.88)
colorLink	Control the color of hyperlink.	string	#1677ff
colorLinkActive	Control the color of hyperlink when clicked.	string	#0958d9
colorLinkHover	Control the color of hyperlink when hovering.	string	#69b1ff
colorPrimary	Brand color is one of the most direct visual elements to reflect the characteristics and communication of the product. After you have selected the brand color, we will automatically generate a complete color palette and assign it effective design semantics.	string	#1677ff
colorSplit	Used as the color of separator, this color is the same as colorBorderSecondary but with transparency.	string	rgba(5, 5, 5, 0.06)
colorText	Default text color which comply with W3C standards, and this color is also the darkest neutral color.	string	rgba(0, 0, 0, 0.88)
colorTextDescription	Control the font color of text description.	string	rgba(0, 0, 0, 0.45)

Token Name	Description	Type	Default Value
colorTextDisabled	Control the color of text in disabled state.	`string`	rgba(0, 0, 0, 0.25)
colorTextHeading	Control the font color of heading.	`string`	rgba(0, 0, 0, 0.88)
colorTextPlaceholder	Control the color of placeholder text.	`string`	rgba(0, 0, 0, 0.25)
borderRadius	Border radius of base components	`number`	6
borderRadiusLG	LG size border radius, used in some large border radius components, such as Card, Modal and other components.	`number`	8
boxShadowSecondary	Control the secondary box shadow style of an element.	`string`	0 6px 16px 0 rgba(0, 0, 0, 0.08), 0 3px 6px -4px rgba(0, 0, 0, 0.12), 0 9px 28px 8px rgba(0, 0, 0, 0.05)
controlHeight	The height of the basic controls such as buttons and input boxes in Ant Design	`number`	32
controlInteractiveSize	Control the interactive size of control component.	`number`	16
controlItemBgActive	Control the background color of control component item when active.	`string`	#e6f4ff
controlItemBgActiveHover	Control the background color of control component item when hovering and active.	`string`	#bae0ff
controlItemBgHover	Control the background color of control component item when hovering.	`string`	rgba(0, 0, 0, 0.04)
fontFamily	The font family of Ant Design prioritizes the default interface font of the system, and provides a set of alternative font libraries that are suitable for screen display to maintain the readability and readability of the font under different platforms and browsers, reflecting the friendly, stable and professional characteristics.	`string`	-apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, 'Helvetica Neue', Arial, 'Noto Sans', sans-serif, 'Apple Color Emoji', 'Segoe UI Emoji', 'Segoe UI Symbol', 'Noto Color Emoji'
fontSize	The most widely used font size in the design system, from which the text gradient will be derived.	`number`	14
fontSizeIcon	Control the font size of operation icon in Select, Cascader, etc. Normally same as fontSizeSM.	`number`	12
fontSizeSM	Small font size	`number`	12
fontWeightStrong	Control the font weight of heading components (such as h1, h2, h3) or selected item.	`number`	600
lineHeight	Line height of text.	`number`	1.5714285714285714

Token Name	Description	Type	Default Value
lineType	Border style of base components	`string`	solid
lineWidth	Border width of base components	`number`	1
margin	Control the margin of an element, with a medium size.	`number`	16
marginXXS	Control the margin of an element, with the smallest size.	`number`	4
motionDurationMid	Motion speed, medium speed. Used for medium element animation interaction.	`string`	0.2s
motionDurationSlow	Motion speed, slow speed. Used for large element animation interaction.	`string`	0.3s
opacityLoading	Control the opacity of the loading state.	`number`	0.65
padding	Control the padding of the element.	`number`	16
paddingContentVerticalLG	Control the vertical padding of content element, suitable for large screen devices.	`number`	16
paddingSM	Control the small padding of the element.	`number`	12
paddingXS	Control the extra small padding of the element.	`number`	8
paddingXXS	Control the extra extra small padding of the element.	`number`	4

Note

According to the [React documentation](), every child in an array should be assigned a unique key. The values inside the Table's `dataSource` and `columns` should follow this rule. By default, `dataSource[i].key` will be treated as the key value for `dataSource`.



If `dataSource[i].key` is not provided, then you should specify the primary key of dataSource value via `rowKey`, as shown below. If not, warnings like the one above will show in browser console.

```
// primary key is uid
return <Table rowKey="uid" />;
// or
return <Table rowKey={(record) => record.uid} />;
```

FAQ

How to hide pagination when single page or no data?

You can set `hideOnSinglePage` with `pagination` prop.

Table will return to first page when filter data.

Table total page count usually reduce after filter data, we by default return to first page in case of current page is out of filtered results.

You may need to keep current page after filtering when fetch data from remote service, please check this demo as workaround.

Also you can use the action from extra param to determine when return to first page.

Why Table pagination show size changer?

In order to improve user experience, Pagination show size changer by default when `total > 50` since `4.1.0`. You can set `showSizeChanger=false` to disable this feature.

Why Table fully render when state change?

Table can not tell what state used in `columns.render`, so it always need fully render to avoid sync issue. You can use `column.shouldCellUpdate` to control render.

How to handle fixed column display over the mask layout?

Fixed column use `z-index` to make it over other columns. You will find sometime fixed columns also over your mask layout. You can set `z-index` on your mask layout to resolve.

How to custom render Table Checkbox（For example, adding Tooltip）?

Since `4.1.0`, You can use `rowSelection.renderCell` to custom render Table Checkbox. If you want to add Tooltip, please refer to this demo.