## zh-CN

通过 `react-window` 引入虚拟滚动方案，实现 100000 条数据的高性能表格。

## en-US

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

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

function VirtualTable(props: Parameters<typeof Table>[0]) {
  const { columns, scroll } = props;
  const [tableWidth, setTableWidth] = useState(0);

  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: () => null,
      set: (scrollLeft: number) => {
        if (gridRef.current) {
          gridRef.current.scrollTo({ scrollLeft });
        }
      },
    });

    return obj;
  });

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

```
  useEffect(() => resetVirtualGrid, [tableWidth]);

  const renderVirtualList = (rawData: object[], { scrollbarSize, ref, onScroll }:
any) => {
    ref.current = connectObject;
    const totalHeight = rawData.length * 54;

    return (
      <Grid
        ref={gridRef}
        className="virtual-grid"
        columnCount={mergedColumns.length}
        columnWidth={(index: number) => {
          const { width } = mergedColumns[index];
          return totalHeight > scroll!.y! && index === mergedColumns.length - 1
            ? (width as number) - scrollbarSize - 1
            : (width as number);
        }}
        height={scroll!.y as number}
        rowCount={rawData.length}
        rowHeight={() => 54}
        width={tableWidth}
        onScroll={({ scrollLeft }: { scrollLeft: number }) => {
          onScroll({ scrollLeft });
        }}
      >
        {({
          columnIndex,
          rowIndex,
          style,
        }: {
          columnIndex: number;
          rowIndex: number;
          style: React.CSSProperties;
        }) => (
          <div
            className={classNames('virtual-table-cell', {
              'virtual-table-cell-last': columnIndex === mergedColumns.length - 1,
            })}
            style={style}
          >
            {(rawData[rowIndex] as any)[(mergedColumns as any)
[columnIndex].dataIndex]}
          </div>
        )}
      </Grid>
    );
  };

  return (
    <ResizeObserver
```

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

// Usage
const columns = [
  { title: 'A', dataIndex: 'key', width: 150 },
  { title: 'B', dataIndex: 'key' },
  { title: 'C', dataIndex: 'key' },
  { title: 'D', dataIndex: 'key' },
  { title: 'E', dataIndex: 'key', width: 200 },
  { title: 'F', dataIndex: 'key', width: 100 },
];

const data = Array.from({ length: 100000 }, (_, key) => ({ key }));

export default () => (
  <VirtualTable columns={columns} dataSource={data} scroll={{ y: 300, x: '100vw' }}
/>
);
```