zh-CN

带单元格编辑功能的表格。当配合 `shouldCellUpdate` 使用时请注意闭包问题。

en-US

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

```tsx
import React, { useContext, useState, useEffect, useRef } from 'react';
import { Table, Input, Button, Popconfirm, Form, InputRef } from 'antd';
import { FormInstance } from 'antd/lib/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)!;

  useEffect(() => {
    if (editing) {
      inputRef.current!.focus();
    }
  }, [editing]);

  const toggleEdit = () => {
    setEditing(!editing);
    form.setFieldsValue({ [dataIndex]: record[dataIndex] });
  };

  const save = async () => {
    try {
      const values = await form.validateFields();

      toggleEdit();
      handleSave({ ...record, ...values });
    } catch (errInfo) {
      console.log('Save failed:', errInfo);
    }
  };

  let childNode = children;

  if (editable) {
    childNode = editing ? (
      <Form.Item
        style={{ margin: 0 }}
        name={dataIndex}
        rules={[
          {
            required: true,
            message: `${title} is required.`,
          },
        ]}
      >
        <Input ref={inputRef} onPressEnter={save} onBlur={save} />
      </Form.Item>
    ) : (
      <div className="editable-cell-value-wrap" style={{ paddingRight: 24 }}
onClick={toggleEdit}>
        {children}
      </div>
    );
  }
  return <td {...restProps}>{childNode}</td>;
};

type EditableTableProps = Parameters<typeof Table>[0];

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

interface EditableTableState {
  dataSource: DataType[];
  count: number;
}

type ColumnTypes = Exclude<EditableTableProps['columns'], undefined>;

class EditableTable extends React.Component<EditableTableProps, EditableTableState>
{
  columns: (ColumnTypes[number] & { editable?: boolean; dataIndex: string })[];

  constructor(props: EditableTableProps) {
    super(props);

    this.columns = [
      {
        title: 'name',
        dataIndex: 'name',
        width: '30%',
        editable: true,
      },
      {
        title: 'age',
        dataIndex: 'age',
      },
      {
        title: 'address',
        dataIndex: 'address',
      },
      {
        title: 'operation',
        dataIndex: 'operation',
        render: (_, record: { key: React.Key }) =>
          this.state.dataSource.length >= 1 ? (
            <Popconfirm title="Sure to delete?" onConfirm={() =>
this.handleDelete(record.key)}>
              <a>Delete</a>
            </Popconfirm>
          ) : null,
      },
    ];

    this.state = {
      dataSource: [
        {
          key: '0',
          name: 'Edward King 0',
          age: '32',
          address: 'London, Park Lane no. 0',
        },
        {
          key: '1',
          name: 'Edward King 1',
          age: '32',
          address: 'London, Park Lane no. 1',
        },
      ],
      count: 2,
    };
  }

  handleDelete = (key: React.Key) => {
    const dataSource = [...this.state.dataSource];
    this.setState({ dataSource: dataSource.filter(item => item.key !== key) });
  };

  handleAdd = () => {
    const { count, dataSource } = this.state;
    const newData: DataType = {
      key: count,
      name: `Edward King ${count}`,
      age: '32',
      address: `London, Park Lane no. ${count}`,
    };
    this.setState({
      dataSource: [...dataSource, newData],
      count: count + 1,
    });
  };

  handleSave = (row: DataType) => {
    const newData = [...this.state.dataSource];
    const index = newData.findIndex(item => row.key === item.key);
    const item = newData[index];
    newData.splice(index, 1, {
      ...item,
      ...row,
    });
    this.setState({ dataSource: newData });
  };
  render() {
    const { dataSource } = this.state;
    const components = {
      body: {
        row: EditableRow,
        cell: EditableCell,
      },
    };
    const columns = this.columns.map(col => {
      if (!col.editable) {
        return col;
      }
      return {
        ...col,
        onCell: (record: DataType) => ({
          record,
          editable: col.editable,
          dataIndex: col.dataIndex,
          title: col.title,
          handleSave: this.handleSave,
        }),
      };
    });
    return (
      <div>
        <Button onClick={this.handleAdd} type="primary" style={{ marginBottom: 16
}}>
          Add a row
        </Button>
        <Table
          components={components}
          rowClassName={() => 'editable-row'}
          bordered
          dataSource={dataSource}
          columns={columns as ColumnTypes}
        />
      </div>
    );
  }
}

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

```
.editable-cell {
  position: relative;
}

.editable-cell-value-wrap {
  padding: 5px 12px;
  cursor: pointer;
}
```

```css
.editable-row:hover .editable-cell-value-wrap {
  padding: 4px 11px;
  border: 1px solid #d9d9d9;
  border-radius: 2px;
}

[data-theme='dark'] .editable-row:hover .editable-cell-value-wrap {
  border: 1px solid #434343;
}
```