zh-CN

将节点拖拽到其他节点内部或前后。

en-US

Drag treeNode to insert after the other treeNode or insert into the other parent TreeNode.

```
import { Tree } from 'antd';

const x = 3;
const y = 2;
const z = 1;
const gData = [];

const generateData = (_level, _preKey, _tns) => {
  const preKey = _preKey || '0';
  const tns = _tns || gData;

  const children = [];
  for (let i = 0; i < x; i++) {
    const key = `${preKey}-${i}`;
    tns.push({ title: key, key });
    if (i < y) {
      children.push(key);
    }
  }
  if (_level < 0) {
    return tns;
  }
  const level = _level - 1;
  children.forEach((key, index) => {
    tns[index].children = [];
    return generateData(level, key, tns[index].children);
  });
};
generateData(z);

class Demo extends React.Component {
  state = {
    gData,
    expandedKeys: ['0-0', '0-0-0', '0-0-0-0'],
  };

  onDragEnter = info => {
    console.log(info);
    // expandedKeys 需要受控时设置
    // this.setState({
    //   expandedKeys: info.expandedKeys,
    // });
  };
```

```
onDrop = info => {
  console.log(info);
  const dropKey = info.node.key;
  const dragKey = info.dragNode.key;
  const dropPos = info.node.pos.split('-');
  const dropPosition = info.dropPosition - Number(dropPos[dropPos.length - 1]);

  const loop = (data, key, callback) => {
    for (let i = 0; i < data.length; i++) {
      if (data[i].key === key) {
        return callback(data[i], i, data);
      }
      if (data[i].children) {
        loop(data[i].children, key, callback);
      }
    }
  };
  const data = [...this.state.gData];

  // Find dragObject
  let dragObj;
  loop(data, dragKey, (item, index, arr) => {
    arr.splice(index, 1);
    dragObj = item;
  });

  if (!info.dropToGap) {
    // Drop on the content
    loop(data, dropKey, item => {
      item.children = item.children || [];
      // where to insert 示例添加到头部，可以是随意位置
      item.children.unshift(dragObj);
    });
  } else if (
    (info.node.props.children || []).length > 0 && // Has children
    info.node.props.expanded && // Is expanded
    dropPosition === 1 // On the bottom gap
  ) {
    loop(data, dropKey, item => {
      item.children = item.children || [];
      // where to insert 示例添加到头部，可以是随意位置
      item.children.unshift(dragObj);
      // in previous version, we use item.children.push(dragObj) to insert the
      // item to the tail of the children
    });
  } else {
    let ar;
    let i;
    loop(data, dropKey, (item, index, arr) => {
      ar = arr;
      i = index;
```

```
      });
      if (dropPosition === -1) {
        ar.splice(i, 0, dragObj);
      } else {
        ar.splice(i + 1, 0, dragObj);
      }
    }

    this.setState({
      gData: data,
    });
  };

  render() {
    return (
      <Tree
        className="draggable-tree"
        defaultExpandedKeys={this.state.expandedKeys}
        draggable
        blockNode
        onDragEnter={this.onDragEnter}
        onDrop={this.onDrop}
        treeData={this.state.gData}
      />
    );
  }
}

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