何时使用 {#when-to-use}

需要用户处理事务，又不希望跳转页面以致打断工作流程时，可以使用 `Modal` 在当前页面正中打开一个浮层，承载相应的操作。

另外当需要一个简洁的确认框询问用户时，可以使用 `App.useApp` 封装的语法糖方法。

代码演示

基本

```
import React, { useState } from 'react';
import { Button, Modal } from 'antd';

const App: React.FC = () => {
  const [isModalOpen, setIsModalOpen] = useState(false);

  const showModal = () => {
    setIsModalOpen(true);
  };

  const handleOk = () => {
    setIsModalOpen(false);
  };

  const handleCancel = () => {
    setIsModalOpen(false);
  };

  return (
    <>
      <Button type="primary" onClick={showModal}>
        Open Modal
      </Button>
      <Modal title="Basic Modal" open={isModalOpen} onOk={handleOk}
onCancel={handleCancel}>
        <p>Some contents...</p>
        <p>Some contents...</p>
        <p>Some contents...</p>
      </Modal>
    </>
  );
};

export default App;
```

异步关闭

```
import React, { useState } from 'react';
import { Button, Modal } from 'antd';

const App: React.FC = () => {
  const [open, setOpen] = useState(false);
  const [confirmLoading, setConfirmLoading] = useState(false);
  const [modalText, setModalText] = useState('Content of the modal');

  const showModal = () => {
    setOpen(true);
  };

  const handleOk = () => {
    setModalText('The modal will be closed after two seconds');
    setConfirmLoading(true);
    setTimeout(() => {
      setOpen(false);
      setConfirmLoading(false);
    }, 2000);
  };

  const handleCancel = () => {
    console.log('Clicked cancel button');
    setOpen(false);
  };

  return (
    <>
      <Button type="primary" onClick={showModal}>
        Open Modal with async logic
      </Button>
      <Modal
        title="Title"
        open={open}
        onOk={handleOk}
        confirmLoading={confirmLoading}
        onCancel={handleCancel}
      >
        <p>{modalText}</p>
      </Modal>
    </>
  );
};
```

```
export default App;
```

自定义页脚

```
import React, { useState } from 'react';
import { Button, Modal } from 'antd';

const App: React.FC = () => {
  const [loading, setLoading] = useState(false);
  const [open, setOpen] = useState(false);

  const showModal = () => {
    setOpen(true);
  };

  const handleOk = () => {
    setLoading(true);
    setTimeout(() => {
      setLoading(false);
      setOpen(false);
    }, 3000);
  };

  const handleCancel = () => {
    setOpen(false);
  };

  return (
    <>
      <Button type="primary" onClick={showModal}>
        Open Modal with customized footer
      </Button>
      <Modal
        open={open}
        title="Title"
        onOk={handleOk}
        onCancel={handleCancel}
        footer={[
          <Button key="back" onClick={handleCancel}>
            Return
          </Button>,
          <Button key="submit" type="primary" loading={loading} onClick=
{handleOk}>
            Submit
          </Button>,
```

```
          <Button
            key="link"
            href="https://google.com"
            target="_blank"
            type="primary"
            loading={loading}
            onClick={handleOk}
          >
            Search on Google
          </Button>,
        ]}
      >
        <p>Some contents...</p>
        <p>Some contents...</p>
        <p>Some contents...</p>
        <p>Some contents...</p>
        <p>Some contents...</p>
      </Modal>
    </>
  );
};

export default App;
```

加载中

v5.18.0

```
import React from 'react';
import { Button, Modal } from 'antd';

const App: React.FC = () => {
  const [open, setOpen] = React.useState<boolean>(false);
  const [loading, setLoading] = React.useState<boolean>(true);

  const showLoading = () => {
    setOpen(true);
    setLoading(true);

    // Simple loading mock. You should add cleanup logic in real world.
    setTimeout(() => {
      setLoading(false);
    }, 2000);
  };

  return (
```

```
    <>
      <Button type="primary" onClick={showLoading}>
        Open Modal
      </Button>
      <Modal
        title={<p>Loading Modal</p>}
        footer={
          <Button type="primary" onClick={showLoading}>
            Reload
          </Button>
        }
        loading={loading}
        open={open}
        onCancel={() => setOpen(false)}
      >
        <p>Some contents...</p>
        <p>Some contents...</p>
        <p>Some contents...</p>
      </Modal>
    </>
  );
};

export default App;
```

自定义页脚渲染函数

v5.9.0

```
import React, { useState } from 'react';
import { Button, Modal, Space } from 'antd';

const App: React.FC = () => {
  const [open, setOpen] = useState(false);

  const showModal = () => {
    setOpen(true);
  };
  const handleOk = () => {
    setOpen(false);
  };

  const handleCancel = () => {
    setOpen(false);
  };
  return (
```

```
    <>
      <Space>
        <Button type="primary" onClick={showModal}>
          Open Modal
        </Button>
        <Button
          type="primary"
          onClick={() => {
            Modal.confirm({
              title: 'Confirm',
              content: 'Bla bla ...',
              footer: (_, { OkBtn, CancelBtn }) => (
                <>
                  <Button>Custom Button</Button>
                  <CancelBtn />
                  <OkBtn />
                </>
              ),
            });
          }}
        >
          Open Modal Confirm
        </Button>
      </Space>
      <Modal
        open={open}
        title="Title"
        onOk={handleOk}
        onCancel={handleCancel}
        footer={(_, { OkBtn, CancelBtn }) => (
          <>
            <Button>Custom Button</Button>
            <CancelBtn />
            <OkBtn />
          </>
        )}
      >
        <p>Some contents...</p>
        <p>Some contents...</p>
        <p>Some contents...</p>
        <p>Some contents...</p>
        <p>Some contents...</p>
      </Modal>
    </>
  );
};
```

```
export default App;
```

使用 hooks 获得上下文

```
import React, { createContext } from 'react';
import { Button, Modal, Space } from 'antd';

const ReachableContext = createContext<string | null>(null);
const UnreachableContext = createContext<string | null>(null);

const config = {
  title: 'Use Hook!',
  content: (
    <>
      <ReachableContext.Consumer>{(name) => `Reachable: ${name}!`}
</ReachableContext.Consumer>
      <br />
      <UnreachableContext.Consumer>{(name) => `Unreachable: ${name}!`}
</UnreachableContext.Consumer>
    </>
  ),
};

const App: React.FC = () => {
  const [modal, contextHolder] = Modal.useModal();

  return (
    <ReachableContext.Provider value="Light">
      <Space>
        <Button
          onClick={async () => {
            const confirmed = await modal.confirm(config);
            console.log('Confirmed: ', confirmed);
          }}
        >
          Confirm
        </Button>
        <Button
          onClick={() => {
            modal.warning(config);
          }}
        >
          Warning
        </Button>
        <Button
```

```
          onClick={async () => {
            modal.info(config);
          }}
        >
          Info
        </Button>
        <Button
          onClick={async () => {
            modal.error(config);
          }}
        >
          Error
        </Button>
      </Space>
      {/* `contextHolder` should always be placed under the context you
want to access */}
      {contextHolder}

      {/* Can not access this context since `contextHolder` is not in it
*/}
      <UnreachableContext.Provider value="Bamboo" />
    </ReachableContext.Provider>
  );
};

export default App;
```

国际化

```
import React, { useState } from 'react';
import { ExclamationCircleOutlined } from '@ant-design/icons';
import { Button, Modal, Space } from 'antd';

const LocalizedModal = () => {
  const [open, setOpen] = useState(false);

  const showModal = () => {
    setOpen(true);
  };

  const hideModal = () => {
    setOpen(false);
  };

  return (
    <>
```

```
      <Button type="primary" onClick={showModal}>
        Modal
      </Button>
      <Modal
        title="Modal"
        open={open}
        onOk={hideModal}
        onCancel={hideModal}
        okText="确认"
        cancelText="取消"
      >
        <p>Bla bla ...</p>
        <p>Bla bla ...</p>
        <p>Bla bla ...</p>
      </Modal>
    </>
  );
};

const App: React.FC = () => {
  const [modal, contextHolder] = Modal.useModal();

  const confirm = () => {
    modal.confirm({
      title: 'Confirm',
      icon: <ExclamationCircleOutlined />,
      content: 'Bla bla ...',
      okText: '确认',
      cancelText: '取消',
    });
  };

  return (
    <>
      <Space>
        <LocalizedModal />
        <Button onClick={confirm}>Confirm</Button>
      </Space>
      {contextHolder}
    </>
  );
};

export default App;
```

手动更新和移除

```tsx
import React from 'react';
import { Button, Modal } from 'antd';

const App: React.FC = () => {
  const [modal, contextHolder] = Modal.useModal();

  const countDown = () => {
    let secondsToGo = 5;

    const instance = modal.success({
      title: 'This is a notification message',
      content: `This modal will be destroyed after ${secondsToGo} second.`,
    });

    const timer = setInterval(() => {
      secondsToGo -= 1;
      instance.update({
        content: `This modal will be destroyed after ${secondsToGo}
second.`,
      });
    }, 1000);

    setTimeout(() => {
      clearInterval(timer);
      instance.destroy();
    }, secondsToGo * 1000);
  };

  return (
    <>
      <Button onClick={countDown}>Open modal to close in 5s</Button>
      {contextHolder}
    </>
  );
};

export default App;
```

自定义位置

```tsx
import React, { useState } from 'react';
import { Button, Modal } from 'antd';

const App: React.FC = () => {
  const [modal1Open, setModal1Open] = useState(false);
```

```
  const [modal2Open, setModal2Open] = useState(false);

  return (
    <>
      <Button type="primary" onClick={() => setModal1Open(true)}>
        Display a modal dialog at 20px to Top
      </Button>
      <Modal
        title="20px to Top"
        style={{ top: 20 }}
        open={modal1Open}
        onOk={() => setModal1Open(false)}
        onCancel={() => setModal1Open(false)}
      >
        <p>some contents...</p>
        <p>some contents...</p>
        <p>some contents...</p>
      </Modal>
      <br />
      <br />
      <Button type="primary" onClick={() => setModal2Open(true)}>
        Vertically centered modal dialog
      </Button>
      <Modal
        title="Vertically centered modal dialog"
        centered
        open={modal2Open}
        onOk={() => setModal2Open(false)}
        onCancel={() => setModal2Open(false)}
      >
        <p>some contents...</p>
        <p>some contents...</p>
        <p>some contents...</p>
      </Modal>
    </>
  );
};

export default App;
```

暗背景

Debug

```
import React, { useState } from 'react';
import { ClockCircleOutlined, DownOutlined } from '@ant-design/icons';
```

```tsx
import {
  Anchor,
  Badge,
  Button,
  Calendar,
  Card,
  Collapse,
  DatePicker,
  Dropdown,
  Modal,
  Slider,
  Switch,
  Table,
  Tabs,
  Timeline,
  Transfer,
  Tree,
  Typography,
} from 'antd';
import type { TableProps, TransferProps } from 'antd';
import type { TransferKey } from 'antd/es/transfer/interface';
import dayjs from 'dayjs';
import customParseFormat from 'dayjs/plugin/customParseFormat';
import difference from 'lodash/difference';

dayjs.extend(customParseFormat);

const { Panel } = Collapse;
const { TreeNode } = Tree;
const { TabPane } = Tabs;
const { Meta } = Card;
const { Link } = Anchor;
const { Text } = Typography;

const text = `
  A dog is a type of domesticated animal.
  Known for its loyalty and faithfulness,
  it can be found as a welcome guest in many households across the world.
`;

interface DataType {
  key: string;
  title: string;
  description: string;
  disabled: boolean;
}
```

```ts
interface RecordType {
  key: string;
  name: string;
  age: number;
  address: string;
}

interface DataTableType {
  key: string;
  name: string;
  borrow: number;
  repayment: number;
}

interface ExpandDataType {
  key: React.Key;
  date: string;
  name: string;
  upgradeNum: string;
}

interface NestDataType {
  key: React.Key;
  name: string;
  platform: string;
  version: string;
  upgradeNum: number;
  creator: string;
  createdAt: string;
}

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

const mockData = Array.from({ length: 20 }).map<DataType>((_, i) => ({
  key: i.toString(),
  title: `content${i + 1}`,
  description: `description of content${i + 1}`,
  disabled: i % 3 < 1,
}));
```

```
const oriTargetKeys = mockData
  .filter((item) => Number(item.key) % 3 > 1)
  .map<TransferKey>((item) => item.key);

const dataSource: RecordType[] = [
  { 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 columnsTable: TableProps<DataTableType>['columns'] = [
  { title: 'Name', dataIndex: 'name' },
  { title: 'Borrow', dataIndex: 'borrow' },
  { title: 'Repayment', dataIndex: 'repayment' },
];

const summaryDataSource: DataTableType[] = [
  { key: '1', name: 'John Brown', borrow: 10, repayment: 33 },
  { key: '2', name: 'Jim Green', borrow: 100, repayment: 0 },
  { key: '3', name: 'Joe Black', borrow: 10, repayment: 10 },
  { key: '4', name: 'Jim Red', borrow: 75, repayment: 45 },
];

const expandDataSource = Array.from({ length: 3 }).map<ExpandDataType>((_,
i) => ({
  key: i,
  date: '2014-12-24 23:12:00',
  name: 'This is production name',
  upgradeNum: 'Upgraded: 56',
}));

const expandColumns: TableProps<ExpandDataType>['columns'] = [
  { title: 'Date', dataIndex: 'date', key: 'date' },
  { title: 'Name', dataIndex: 'name', key: 'name' },
  {
    title: 'Status',
    key: 'state',
    render: () => (
      <span>
        <Badge status="success" />
        Finished
```

```
      </span>
    ),
  },
  { title: 'Upgrade Status', dataIndex: 'upgradeNum', key: 'upgradeNum' },
  {
    title: 'Action',
    dataIndex: 'operation',
    key: 'operation',
    render: () => (
      <span className="table-operation">
        <a>Pause</a>
        <a>Stop</a>
        <Dropdown>
          <a>
            More <DownOutlined />
          </a>
        </Dropdown>
      </span>
    ),
  },
];

const expandedRowRender = () => (
  <Table<ExpandDataType> columns={expandColumns} dataSource=
{expandDataSource} pagination={false} />
);

const columnsNest: TableProps<NestDataType>['columns'] = [
  { title: 'Name', dataIndex: 'name', key: 'name' },
  { title: 'Platform', dataIndex: 'platform', key: 'platform' },
  { title: 'Version', dataIndex: 'version', key: 'version' },
  { title: 'Upgraded', dataIndex: 'upgradeNum', key: 'upgradeNum' },
  { title: 'Creator', dataIndex: 'creator', key: 'creator' },
  { title: 'Date', dataIndex: 'createdAt', key: 'createdAt' },
  { title: 'Action', key: 'operation', render: () => <a>Publish</a> },
];

const nestDataSource = Array.from({ length: 3 }).map<NestDataType>((_, i)
=> ({
  key: i,
  name: 'Screem',
  platform: 'iOS',
  version: '10.3.4.5654',
  upgradeNum: 500,
  creator: 'Jack',
  createdAt: '2014-12-24 23:12:00',
```

```
}));

const columnsFixed: TableProps<FixedDataType>['columns'] = [
  { 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' },
  { 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 fixedDataSource: FixedDataType[] = [
  { key: '1', name: 'John Brown', age: 32, address: 'New York Park' },
  { key: '2', name: 'Jim Green', age: 40, address: 'London Park' },
];

const TableTransfer: React.FC<
  Readonly<Partial<Record<'leftColumns' | 'rightColumns',
TableProps<DataType>['columns']>>> &
    TransferProps<DataType>
> = (props) => {
  const { leftColumns, rightColumns, ...restProps } = props;
  return (
    <Transfer<DataType> {...restProps} showSelectAll={false}>
      {(transferProps) => {
        const {
          direction,
          filteredItems,
          onItemSelectAll,
          onItemSelect,
          selectedKeys: listSelectedKeys,
          disabled: listDisabled,
        } = transferProps;

        const columns = (direction === 'left' ? leftColumns : rightColumns)
?? [];

        const rowSelection: TableProps<DataType>['rowSelection'] = {
```

```
          getCheckboxProps: (item) => ({ disabled: listDisabled ||
item.disabled }),
          onSelectAll(selected, selectedRows) {
            const treeSelectedKeys = selectedRows
              .filter((item) => !item.disabled)
              .map(({ key }) => key);
            const diffKeys = selected
              ? difference(treeSelectedKeys, listSelectedKeys)
              : difference(listSelectedKeys, treeSelectedKeys);
            onItemSelectAll(diffKeys, selected);
          },
          onSelect({ key }, selected) {
            onItemSelect(key, selected);
          },
          selectedRowKeys: listSelectedKeys,
        };

        return (
          <Table<DataType>
            id="components-transfer-table"
            rowSelection={rowSelection}
            columns={columns}
            dataSource={filteredItems}
            size="small"
            style={{ pointerEvents: listDisabled ? 'none' : 'auto' }}
            onRow={({ key, disabled: itemDisabled }) => ({
              onClick: () => {
                if (itemDisabled || listDisabled) {
                  return;
                }
                onItemSelect(key, !listSelectedKeys.includes(key));
              },
            })}
          />
        );
      }}
    </Transfer>
  );
};

const columns: TableProps<RecordType>['columns'] = [
  {
    title: 'Name',
    dataIndex: 'name',
    key: 'name',
    filters: [
```

```
      { text: 'Joe', value: 'Joe' },
      { text: 'Jim', value: 'Jim' },
    ],
    filteredValue: null,
    onFilter: (value, record) => record.name.includes(value as string),
    sorter: (a, b) => a.name.length - b.name.length,
    sortOrder: 'ascend',
    ellipsis: true,
  },
  {
    title: 'Age',
    dataIndex: 'age',
    key: 'age',
    sorter: false,
    sortOrder: 'ascend',
    ellipsis: true,
  },
  {
    title: 'Address',
    dataIndex: 'address',
    key: 'address',
    filters: [
      { text: 'London', value: 'London' },
      { text: 'New York', value: 'New York' },
    ],
    filteredValue: null,
    onFilter: (value, record) => record.address.includes(value as string),
    sorter: false,
    sortOrder: 'ascend',
    ellipsis: true,
  },
];

const tableTransferColumns: TableProps<DataType>['columns'] = [
  { dataIndex: 'title', title: 'Name' },
  { dataIndex: 'description', title: 'Description' },
];

const Demo: React.FC = () => {
  const [open, setOpen] = useState(false);
  const [targetKeys, setTargetKeys] = useState<TransferKey[]>
(oriTargetKeys);
  const [selectedKeys, setSelectedKeys] = useState<TransferKey[]>([]);
  const [disabled, setDisabled] = useState(false);
  const [showSearch, setShowSearch] = useState(false);
```

```tsx
  const handleDisable = (isDisabled: boolean) => {
    setDisabled(isDisabled);
  };

  const handleTableTransferChange = (nextTargetKeys: TransferKey[]) => {
    setTargetKeys(nextTargetKeys);
  };

  const triggerDisable = (isDisabled: boolean) => {
    setDisabled(isDisabled);
  };

  const triggerShowSearch = (isShowSearch: boolean) => {
    setShowSearch(isShowSearch);
  };

  const handleTransferChange = (keys: TransferKey[]) => {
    setTargetKeys(keys);
  };

  const handleTransferSelectChange = (
    sourceSelectedKeys: TransferKey[],
    targetSelectedKeys: TransferKey[],
  ) => {
    setSelectedKeys([...sourceSelectedKeys, ...targetSelectedKeys]);
  };

  const showModal = () => {
    setOpen(true);
  };

  const handleOk = (e: React.MouseEvent<HTMLButtonElement, MouseEvent>) =>
{
    console.log(e);
    setOpen(false);
  };

  const handleCancel = (e: React.MouseEvent<HTMLButtonElement, MouseEvent>)
=> {
    console.log(e);
    setOpen(false);
  };

  return (
    <>
      <Button type="primary" onClick={showModal}>
```

```
        Open Modal
      </Button>
      <Modal title="Basic Modal" open={open} onOk={handleOk} onCancel=
{handleCancel}>
        <Switch
          unCheckedChildren="disabled"
          checkedChildren="disabled"
          checked={disabled}
          onChange={handleDisable}
          style={{ marginBottom: 16 }}
        />
        <Card title="Card Title">
          <Card.Grid>Content</Card.Grid>
          <Card.Grid hoverable={false}>Content</Card.Grid>
          <Card.Grid>Content</Card.Grid>
          <Card.Grid>Content</Card.Grid>
          <Card.Grid>Content</Card.Grid>
          <Card.Grid>Content</Card.Grid>
          <Card.Grid>Content</Card.Grid>
        </Card>
        <Collapse>
          <Panel header="This is panel header 1" key="1">
            <Collapse defaultActiveKey="1">
              <Panel header="This is panel nest panel" key="1">
                <p>{text}</p>
              </Panel>
            </Collapse>
          </Panel>
          <Panel header="This is panel header 2" key="2">
            <p>{text}</p>
          </Panel>
          <Panel header="This is panel header 3" key="3">
            <p>{text}</p>
          </Panel>
        </Collapse>
        <Transfer<DataType>
          dataSource={mockData}
          titles={['Source', 'Target']}
          targetKeys={targetKeys}
          selectedKeys={selectedKeys}
          onChange={handleTransferChange}
          onSelectChange={handleTransferSelectChange}
          render={(item) => item.title}
          disabled={disabled}
        />
        <TableTransfer
```

```
          dataSource={mockData}
          targetKeys={targetKeys}
          disabled={disabled}
          showSearch={showSearch}
          leftColumns={tableTransferColumns}
          rightColumns={tableTransferColumns}
          onChange={handleTableTransferChange}
          filterOption={(inputValue: string, item: any) =>
            item.title?.includes(inputValue) ||
item.tag?.includes(inputValue)
          }
        />
        <Switch
          unCheckedChildren="disabled"
          checkedChildren="disabled"
          checked={disabled}
          onChange={triggerDisable}
          style={{ marginTop: 16 }}
        />
        <Switch
          unCheckedChildren="showSearch"
          checkedChildren="showSearch"
          checked={showSearch}
          onChange={triggerShowSearch}
          style={{ marginTop: 16 }}
        />
        <Anchor>
          <Link href="#anchor-demo-basic" title="Basic demo" />
          <Link href="#anchor-demo-static" title="Static demo" />
          <Link href="#anchor-demo-basic" title="Basic demo with Target"
target="_blank" />
          <Link href="#API" title="API">
            <Link href="#Anchor-Props" title="Anchor Props" />
            <Link href="#Link-Props" title="Link Props" />
          </Link>
        </Anchor>
        <Tabs type="card">
          <TabPane tab="Tab 1" key="1">
            Content of Tab Pane 1
          </TabPane>
          <TabPane tab="Tab 2" key="2">
            Content of Tab Pane 2
          </TabPane>
          <TabPane tab="Tab 3" key="3">
            Content of Tab Pane 3
          </TabPane>
```

```
        </Tabs>
        <Timeline>
          <Timeline.Item>Create a services site 2015-09-01</Timeline.Item>
          <Timeline.Item>Solve initial network problems 2015-09-
01</Timeline.Item>
          <Timeline.Item dot={<ClockCircleOutlined style={{ fontSize:
'16px' }} />} color="red">
            Technical testing 2015-09-01
          </Timeline.Item>
          <Timeline.Item>Network problems being solved 2015-09-
01</Timeline.Item>
        </Timeline>
        <Calendar />
        <Tree showLine switcherIcon={<DownOutlined />} defaultExpandedKeys=
{['0-0-0']}>
          <TreeNode title="parent 1" key="0-0">
            <TreeNode title="parent 1-0" key="0-0-0">
              <TreeNode title="leaf" key="0-0-0-0" />
              <TreeNode title="leaf" key="0-0-0-1" />
              <TreeNode title="leaf" key="0-0-0-2" />
            </TreeNode>
            <TreeNode title="parent 1-1" key="0-0-1">
              <TreeNode title="leaf" key="0-0-1-0" />
            </TreeNode>
            <TreeNode title="parent 1-2" key="0-0-2">
              <TreeNode title="leaf" key="0-0-2-0" />
              <TreeNode title="leaf" key="0-0-2-1" />
            </TreeNode>
          </TreeNode>
        </Tree>
        <Table<RecordType> columns={columns} dataSource={dataSource}
footer={() => 'Footer'} />
        <Table<DataTableType>
          columns={columnsTable}
          dataSource={summaryDataSource}
          pagination={false}
          id="table-demo-summary"
          bordered
          summary={(pageData) => {
            let totalBorrow = 0;
            let totalRepayment = 0;
            pageData.forEach(({ borrow, repayment }) => {
              totalBorrow += borrow;
              totalRepayment += repayment;
            });
            return (
```

```
              <>
                <tr>
                  <th>Total</th>
                  <td>
                    <Text type="danger">{totalBorrow}</Text>
                  </td>
                  <td>
                    <Text>{totalRepayment}</Text>
                  </td>
                </tr>
                <tr>
                  <th>Balance</th>
                  <td colSpan={2}>
                    <Text type="danger">{totalBorrow - totalRepayment}
</Text>
                  </td>
                </tr>
              </>
            );
          }}
        />
        <br />
        <Table<NestDataType>
          columns={columnsNest}
          expandable={{ expandedRowRender }}
          dataSource={nestDataSource}
        />
        <Table<FixedDataType>
          columns={columnsFixed}
          dataSource={fixedDataSource}
          scroll={{ x: 1300, y: 100 }}
        />
        <Card
          hoverable
          style={{ width: 240 }}
          cover={
            <img
              draggable={false}
              alt="example"

src="https://os.alipayobjects.com/rmsportal/QBnOOoLaAfKPirc.png"
              />
          }
        >
          <Meta title="Europe Street beat" description="www.instagram.com"
/>
```

```
        </Card>
        <Slider defaultValue={30} />
        <DatePicker defaultValue={dayjs('2015/01/01', 'YYYY/MM/DD')}
format="YYYY/MM/DD" />
        <Badge count={5}>
          <a href="#" className="head-example" />
        </Badge>
      </Modal>
    </>
  );
};

export default Demo;
```

自定义页脚按钮属性

```
import React, { useState } from 'react';
import { Button, Modal } from 'antd';

const App: React.FC = () => {
  const [open, setOpen] = useState(false);

  const showModal = () => {
    setOpen(true);
  };

  const handleOk = (e: React.MouseEvent<HTMLElement>) => {
    console.log(e);
    setOpen(false);
  };

  const handleCancel = (e: React.MouseEvent<HTMLElement>) => {
    console.log(e);
    setOpen(false);
  };

  return (
    <>
      <Button type="primary" onClick={showModal}>
        Open Modal with customized button props
      </Button>
      <Modal
        title="Basic Modal"
        open={open}
        onOk={handleOk}
        onCancel={handleCancel}
```

```
        okButtonProps={{ disabled: true }}
        cancelButtonProps={{ disabled: true }}
      >
        <p>Some contents...</p>
        <p>Some contents...</p>
        <p>Some contents...</p>
      </Modal>
    </>
  );
};

export default App;
```

自定义渲染对话框

```
import React, { useRef, useState } from 'react';
import { Button, Modal } from 'antd';
import type { DraggableData, DraggableEvent } from 'react-draggable';
import Draggable from 'react-draggable';

const App: React.FC = () => {
  const [open, setOpen] = useState(false);
  const [disabled, setDisabled] = useState(true);
  const [bounds, setBounds] = useState({ left: 0, top: 0, bottom: 0, right:
0 });
  const draggleRef = useRef<HTMLDivElement>(null!);

  const showModal = () => {
    setOpen(true);
  };

  const handleOk = (e: React.MouseEvent<HTMLElement>) => {
    console.log(e);
    setOpen(false);
  };

  const handleCancel = (e: React.MouseEvent<HTMLElement>) => {
    console.log(e);
    setOpen(false);
  };

  const onStart = (_event: DraggableEvent, uiData: DraggableData) => {
    const { clientWidth, clientHeight } = window.document.documentElement;
    const targetRect = draggleRef.current?.getBoundingClientRect();
    if (!targetRect) {
      return;
```

```
    }
    setBounds({
      left: -targetRect.left + uiData.x,
      right: clientWidth - (targetRect.right - uiData.x),
      top: -targetRect.top + uiData.y,
      bottom: clientHeight - (targetRect.bottom - uiData.y),
    });
  };

  return (
    <>
      <Button onClick={showModal}>Open Draggable Modal</Button>
      <Modal
        title={
          <div
            style={{ width: '100%', cursor: 'move' }}
            onMouseOver={() => {
              if (disabled) {
                setDisabled(false);
              }
            }}
            onMouseOut={() => {
              setDisabled(true);
            }}
            // fix eslintjsx-a11y/mouse-events-have-key-events
            // https://github.com/jsx-eslint/eslint-plugin-jsx-
a11y/blob/master/docs/rules/mouse-events-have-key-events.md
            onFocus={() => {}}
            onBlur={() => {}}
            // end
          >
            Draggable Modal
          </div>
        }
        open={open}
        onOk={handleOk}
        onCancel={handleCancel}
        modalRender={(modal) => (
          <Draggable
            disabled={disabled}
            bounds={bounds}
            nodeRef={draggleRef}
            onStart={(event, uiData) => onStart(event, uiData)}
          >
            <div ref={draggleRef}>{modal}</div>
          </Draggable>
```

```
        )}
      >
        <p>
          Just don&apos;t learn physics at school and your life will be
full of magic and miracles.
        </p>
        <br />
        <p>Day before yesterday I saw a rabbit, and yesterday a deer, and
today, you.</p>
      </Modal>
    </>
  );
};

export default App;
```

自定义模态的宽度

```
import React, { useState } from 'react';
import { Button, Flex, Modal } from 'antd';

const App: React.FC = () => {
  const [open, setOpen] = useState(false);
  const [openResponsive, setOpenResponsive] = useState(false);

  return (
    <Flex vertical gap="middle" align="flex-start">
      {/* Basic */}
      <Button type="primary" onClick={() => setOpen(true)}>
        Open Modal of 1000px width
      </Button>
      <Modal
        title="Modal 1000px width"
        centered
        open={open}
        onOk={() => setOpen(false)}
        onCancel={() => setOpen(false)}
        width={1000}
      >
        <p>some contents...</p>
        <p>some contents...</p>
        <p>some contents...</p>
      </Modal>

      {/* Responsive */}
      <Button type="primary" onClick={() => setOpenResponsive(true)}>
```

```
        Open Modal of responsive width
      </Button>
      <Modal
        title="Modal responsive width"
        centered
        open={openResponsive}
        onOk={() => setOpenResponsive(false)}
        onCancel={() => setOpenResponsive(false)}
        width={{
          xs: '90%',
          sm: '80%',
          md: '70%',
          lg: '60%',
          xl: '50%',
          xxl: '40%',
        }}
      >
        <p>some contents...</p>
        <p>some contents...</p>
        <p>some contents...</p>
      </Modal>
    </Flex>
  );
};

export default App;
```

静态方法

```
import React from 'react';
import { Button, Modal, Space } from 'antd';

const info = () => {
  Modal.info({
    title: 'This is a notification message',
    content: (
      <div>
        <p>some messages...some messages...</p>
        <p>some messages...some messages...</p>
      </div>
    ),
    onOk() {},
  });
};

const success = () => {
```

```
  Modal.success({
    content: 'some messages...some messages...',
  });
};

const error = () => {
  Modal.error({
    title: 'This is an error message',
    content: 'some messages...some messages...',
  });
};

const warning = () => {
  Modal.warning({
    title: 'This is a warning message',
    content: 'some messages...some messages...',
  });
};

const App: React.FC = () => (
  <Space wrap>
    <Button onClick={info}>Info</Button>
    <Button onClick={success}>Success</Button>
    <Button onClick={error}>Error</Button>
    <Button onClick={warning}>Warning</Button>
  </Space>
);

export default App;
```

静态确认对话框

```
import React from 'react';
import { ExclamationCircleFilled } from '@ant-design/icons';
import { Button, Modal, Space } from 'antd';

const { confirm } = Modal;

const showConfirm = () => {
  confirm({
    title: 'Do you want to delete these items?',
    icon: <ExclamationCircleFilled />,
    content: 'Some descriptions',
    onOk() {
      console.log('OK');
    },
```

```
    onCancel() {
      console.log('Cancel');
    },
  });
};

const showPromiseConfirm = () => {
  confirm({
    title: 'Do you want to delete these items?',
    icon: <ExclamationCircleFilled />,
    content: 'When clicked the OK button, this dialog will be closed after
1 second',
    onOk() {
      return new Promise((resolve, reject) => {
        setTimeout(Math.random() > 0.5 ? resolve : reject, 1000);
      }).catch(() => console.log('Oops errors!'));
    },
    onCancel() {},
  });
};

const showDeleteConfirm = () => {
  confirm({
    title: 'Are you sure delete this task?',
    icon: <ExclamationCircleFilled />,
    content: 'Some descriptions',
    okText: 'Yes',
    okType: 'danger',
    cancelText: 'No',
    onOk() {
      console.log('OK');
    },
    onCancel() {
      console.log('Cancel');
    },
  });
};

const showPropsConfirm = () => {
  confirm({
    title: 'Are you sure delete this task?',
    icon: <ExclamationCircleFilled />,
    content: 'Some descriptions',
    okText: 'Yes',
    okType: 'danger',
    okButtonProps: {
```

```
      disabled: true,
    },
    cancelText: 'No',
    onOk() {
      console.log('OK');
    },
    onCancel() {
      console.log('Cancel');
    },
  });
};

const App: React.FC = () => (
  <Space wrap>
    <Button onClick={showConfirm}>Confirm</Button>
    <Button onClick={showPromiseConfirm}>With promise</Button>
    <Button onClick={showDeleteConfirm} type="dashed">
      Delete
    </Button>
    <Button onClick={showPropsConfirm} type="dashed">
      With extra props
    </Button>
  </Space>
);

export default App;
```

自定义内部模块 className

```
import React, { useState } from 'react';
import { Button, ConfigProvider, Modal, Space } from 'antd';
import { createStyles, useTheme } from 'antd-style';

const useStyle = createStyles(({ token }) => ({
  'my-modal-body': {
    background: token.blue1,
    padding: token.paddingSM,
  },
  'my-modal-mask': {
    boxShadow: `inset 0 0 15px #fff`,
  },
  'my-modal-header': {
    borderBottom: `1px dotted ${token.colorPrimary}`,
  },
  'my-modal-footer': {
    color: token.colorPrimary,
```

```
  },
  'my-modal-content': {
    border: '1px solid #333',
  },
}));

const App: React.FC = () => {
  const [isModalOpen, setIsModalOpen] = useState([false, false]);
  const { styles } = useStyle();
  const token = useTheme();

  const toggleModal = (idx: number, target: boolean) => {
    setIsModalOpen((p) => {
      p[idx] = target;
      return [...p];
    });
  };

  const classNames = {
    body: styles['my-modal-body'],
    mask: styles['my-modal-mask'],
    header: styles['my-modal-header'],
    footer: styles['my-modal-footer'],
    content: styles['my-modal-content'],
  };

  const modalStyles = {
    header: {
      borderLeft: `5px solid ${token.colorPrimary}`,
      borderRadius: 0,
      paddingInlineStart: 5,
    },
    body: {
      boxShadow: 'inset 0 0 5px #999',
      borderRadius: 5,
    },
    mask: {
      backdropFilter: 'blur(10px)',
    },
    footer: {
      borderTop: '1px solid #333',
    },
    content: {
      boxShadow: '0 0 30px #999',
    },
  };
```

```
  return (
    <>
      <Space>
        <Button type="primary" onClick={() => toggleModal(0, true)}>
          Open Modal
        </Button>
        <Button type="primary" onClick={() => toggleModal(1, true)}>
          ConfigProvider
        </Button>
      </Space>
      <Modal
        title="Basic Modal"
        open={isModalOpen[0]}
        onOk={() => toggleModal(0, false)}
        onCancel={() => toggleModal(0, false)}
        footer="Footer"
        classNames={classNames}
        styles={modalStyles}
      >
        <p>Some contents...</p>
        <p>Some contents...</p>
        <p>Some contents...</p>
      </Modal>
      <ConfigProvider
        modal={{
          classNames,
          styles: modalStyles,
        }}
      >
        <Modal
          title="Basic Modal"
          open={isModalOpen[1]}
          onOk={() => toggleModal(1, false)}
          onCancel={() => toggleModal(1, false)}
          footer="Footer"
        >
          <p>Some contents...</p>
          <p>Some contents...</p>
          <p>Some contents...</p>
        </Modal>
      </ConfigProvider>
    </>
  );
};
```

```
export default App;
```

销毁确认对话框

```
import React from 'react';
import { ExclamationCircleOutlined } from '@ant-design/icons';
import { Button, Modal } from 'antd';

const { confirm } = Modal;

const destroyAll = () => {
  Modal.destroyAll();
};

const showConfirm = () => {
  for (let i = 0; i < 3; i += 1) {
    setTimeout(() => {
      confirm({
        icon: <ExclamationCircleOutlined />,
        content: <Button onClick={destroyAll}>Click to destroy
all</Button>,
        onOk() {
          console.log('OK');
        },
        onCancel() {
          console.log('Cancel');
        },
      });
    }, i * 500);
  }
};

const App: React.FC = () => <Button onClick={showConfirm}>Confirm</Button>;

export default App;
```

嵌套弹框

Debug

```
import React, { useState } from 'react';
import { Button, message, Modal, notification, Select, Space, Switch } from
'antd';

const options = [
```

```
  {
    label: 'Option 1',
    value: '1',
  },
  {
    label: 'Option 2',
    value: '2',
  },
];

const Demo: React.FC = () => {
  const [messageInstance, messageHolder] = message.useMessage();
  const [notificationInstance, notificationHolder] =
notification.useNotification();

  const [isModalOpen, setIsModalOpen] = useState<boolean>(false);

  const onShowStatic = () => {
    Modal.confirm({
      content: <Select open value="1" options={options} />,
    });
  };

  return (
    <Space>
      <Switch
        style={{ position: 'relative', zIndex: isModalOpen ? 4000 : 0 }}
        checkedChildren="Open"
        unCheckedChildren="Close"
        onChange={(open) => setIsModalOpen(open)}
      />
      <Button onClick={onShowStatic}>Static</Button>
      <Modal
        title="Basic Modal"
        open={isModalOpen}
        footer={null}
        destroyOnClose
        onCancel={() => setIsModalOpen(false)}
        maskClosable={false}
        closable={false}
        styles={{
          content: {
            marginBlockStart: 100,
          },
        }}
      >
```

```
<Select open value="1" options={options} />
<Modal
  title="Nested Modal"
  open={isModalOpen}
  footer={null}
  destroyOnClose
  mask={false}
  onCancel={() => setIsModalOpen(false)}
  maskClosable={false}
  closable={false}
  styles={{
    content: {
      marginBlockStart: 250,
    },
    body: {
      display: 'flex',
      justifyContent: 'center',
    },
  }}
>
  <Select open value="1" options={options} />

  <Modal
    title="Nested Modal"
    open={isModalOpen}
    footer={null}
    destroyOnClose
    mask={false}
    maskClosable={false}
    onCancel={() => setIsModalOpen(false)}
    closable={false}
    styles={{
      content: {
        marginBlockStart: 400,
      },
      body: {
        display: 'flex',
        justifyContent: 'flex-end',
      },
    }}
  >
    <Space wrap>
      <Button
        onClick={() => {
          Modal.confirm({
            title: 'Are you OK?',
```

```
                content: 'I am OK',
              });
            }}
          >
            Static Confirm
          </Button>

          <Button
            onClick={() => {
              message.success('Hello World');
              notification.success({
                message: 'Hello World',
              });
            }}
          >
            Static Message, Notification
          </Button>

          <Button
            onClick={() => {
              messageInstance.success('Hello World');
              notificationInstance.success({
                message: 'Hello World',
              });
            }}
          >
            Hook Message, Notification
          </Button>

          <Select open value="1" options={options} />
        </Space>
      </Modal>
    </Modal>
  </Modal>

  {messageHolder}
  {notificationHolder}
</Space>
);
};

export default Demo;
```

_InternalPanelDoNotUseOrYouWillBeFired

Debug

```tsx
import React from 'react';
import { Button, Modal, Space, Typography } from 'antd';
import type { ModalFuncProps } from 'antd';

/** Test usage. Do not use in your production. */
const { _InternalPanelDoNotUseOrYouWillBeFired: InternalPanel } = Modal;

const customFooterFn: ModalFuncProps['footer'] = (originNode, { OkBtn,
CancelBtn }) => (
  <Space direction="vertical">
    <Space>{originNode}</Space>
    <Space>
      <CancelBtn />
      <Button danger type="primary">
        Custom
      </Button>
      <OkBtn />
    </Space>
  </Space>
);

export default () => (
  <div style={{ display: 'flex', flexDirection: 'column', rowGap: 16 }}>
    <InternalPanel title="Hello World!" style={{ width: '100%', height: 200
}}>
      Hello World?!
    </InternalPanel>
    <InternalPanel type="success" style={{ width: 200, height: 150 }}>
      A good news!
    </InternalPanel>
    <InternalPanel title="Confirm This?" type="confirm" style={{ width:
300, height: 200 }}>
      Some descriptions.
    </InternalPanel>

    <InternalPanel
      title="Custom Footer Render"
      style={{ width: 380, height: 200 }}
      footer={customFooterFn}
    >
      <Typography.Paragraph>
        <Typography.Link href="https://github.com/ant-design/ant-
design/pull/44318">
          Feature #44318
        </Typography.Link>
      </Typography.Paragraph>
```

```
    </InternalPanel>
  </div>
);
```

控制弹框动画原点

Debug

```
import React, { useState } from 'react';
import { Button, Modal } from 'antd';

const App: React.FC = () => {
  const [isModalOpen, setIsModalOpen] = useState(false);

  const showModal = () => {
    setIsModalOpen(true);
  };

  const handleOk = () => {
    setIsModalOpen(false);
  };

  const handleCancel = () => {
    setIsModalOpen(false);
  };

  return (
    <>
      <Button type="primary" onClick={showModal}>
        Open Modal
      </Button>
      <Modal
        title="Basic Modal"
        open={isModalOpen}
        onOk={handleOk}
        onCancel={handleCancel}
        mousePosition={{ x: 300, y: 300 }}
      >
        <p>Some contents...</p>
        <p>Some contents...</p>
        <p>Some contents...</p>
      </Modal>
    </>
  );
};
```

```
export default App;
```

线框风格

Debug

```
import React from 'react';
import { ConfigProvider, Modal } from 'antd';

/** Test usage. Do not use in your production. */
const { _InternalPanelDoNotUseOrYouWillBeFired: InternalPanel } = Modal;

export default () => (
  <ConfigProvider theme={{ token: { wireframe: true } }}>
    <div style={{ display: 'flex', flexDirection: 'column', rowGap: 16 }}>
      <InternalPanel title="Hello World!" style={{ width: '100%' }}>
        Hello World?!
      </InternalPanel>
      <InternalPanel type="success" style={{ width: 200 }}>
        A good news!
      </InternalPanel>
      <InternalPanel title="Confirm This?" type="confirm" style={{ width:
300 }}>
        Some descriptions.
      </InternalPanel>
    </div>
  </ConfigProvider>
);
```

组件 Token

Debug

```
import React from 'react';
import { ConfigProvider, Modal } from 'antd';

/** Test usage. Do not use in your production. */
const { _InternalPanelDoNotUseOrYouWillBeFired: InternalPanel } = Modal;

export default () => (
  <ConfigProvider
    theme={{
      components: {
        Modal: {
          footerBg: '#f6ffed',
```

```
          contentBg: '#e6fffb',
          headerBg: '#f9f0ff',
          titleLineHeight: 3,
          titleFontSize: 12,
          titleColor: '#1d39c4',
        },
      },
    }}
  >
    <div style={{ display: 'flex', flexDirection: 'column', rowGap: 16 }}>
      <InternalPanel title="Hello World!" style={{ width: '100%' }}>
        Hello World?!
      </InternalPanel>
      <ConfigProvider theme={{ token: { wireframe: true } }}>
        <InternalPanel title="Hello World!" style={{ width: '100%' }}>
          Hello World?!
        </InternalPanel>
      </ConfigProvider>
      <InternalPanel type="success" style={{ width: 200 }}>
        A good news!
      </InternalPanel>
      <InternalPanel title="Confirm This?" type="confirm" style={{ width:
300 }}>
        Some descriptions.
      </InternalPanel>
    </div>
  </ConfigProvider>
);
```

API

通用属性参考：通用属性

参数	说明	类型	默认值	版本
afterClose	Modal 完全关闭后的回调	function	-	
classNames	配置弹窗内置模块的 className	Record<SemanticDOM, string>	-	
styles	配置弹窗内置模块的 style	Record<SemanticDOM, CSSProperties>	-	5.10.0

cancelButtonProps	cancel 按钮 props	ButtonProps	-	
cancelText	取消按钮文字	ReactNode	取消	
centered	垂直居中展示 Modal	boolean	false	
closable	是否显示右上角的关闭按钮	boolean | { closeIcon?: React.ReactNode; disabled?: boolean; }	true	
closeIcon	自定义关闭图标。5.7.0：设置为 `null` 或 `false` 时隐藏关闭按钮	ReactNode	<CloseOutlined />	
confirmLoading	确定按钮 loading	boolean	false	
destroyOnClose	关闭时销毁 Modal 里的子元素	boolean	false	
focusTriggerAfterClose	对话框关闭后是否需要聚焦触发元素	boolean	true	4.9.0
footer	底部内容，当不需要默认底部按钮时，可以设为 `footer={null}`	ReactNode | (originNode: ReactNode, extra: { OkBtn: React.FC, CancelBtn: React.FC }) => ReactNode	(确定取消按钮)	renderFu 5.9.0
forceRender	强制渲染 Modal	boolean	false	
getContainer	指定 Modal 挂载的节点，但依旧为全屏展示，	HTMLElement | () => HTMLElement | Selectors | false	document.body	

	`false` 为挂载在当前位置			
keyboard	是否支持键盘 esc 关闭	boolean	true	
mask	是否展示遮罩	boolean	true	
maskClosable	点击蒙层是否允许关闭	boolean	true	
modalRender	自定义渲染对话框	(node: ReactNode) => ReactNode	-	4.7.0
okButtonProps	ok 按钮 props	ButtonProps	-	
okText	确认按钮文字	ReactNode	确定	
okType	确认按钮类型	string	`primary`	
style	可用于设置浮层的样式，调整浮层位置等	CSSProperties	-	
loading	显示骨架屏	boolean		5.18.0
title	标题	ReactNode	-	
open	对话框是否可见	boolean	-	
width	宽度	string | number | Breakpoint	520	Breakpoi 5.23.0
wrapClassName	对话框外层容器的类名	string	-	
zIndex	设置 Modal 的 `z-index`	number	1000	

onCancel	点击遮罩层或右上角叉或取消按钮的回调	function(e)	-	
onOk	点击确定回调	function(e)	-	
afterOpenChange	打开和关闭 Modal 时动画结束后的回调	(open: boolean) => void	-	5.4.0

注意

- `<Modal />` 默认关闭后状态不会自动清空，如果希望每次打开都是新内容，请设置 `destroyOnClose`。
- `<Modal />` 和 Form 一起配合使用时，设置 `destroyOnClose` 也不会在 Modal 关闭时销毁表单字段数据，需要设置 `<Form preserve={false} />`。
- `Modal.method()` RTL 模式仅支持 hooks 用法。

Modal.method()

包括：

- `Modal.info`
- `Modal.success`
- `Modal.error`
- `Modal.warning`
- `Modal.confirm`

以上均为一个函数，参数为 object，具体属性如下：

参数	说明	类型	默认值	版本
afterClose	Modal 完全关闭后的回调	function	-	4.9.0
autoFocusButton	指定自动获得焦点的按钮	null | ok | cancel	ok	
cancelButtonProps	cancel 按钮 props	ButtonProps	-	
cancelText	设置 Modal.confirm 取消按钮文字	string	取消	
centered	垂直居中展示 Modal	boolean	false	

className	容器类名	string	-	
closable	是否显示右上角的关闭按钮	boolean	false	4.9.0
closeIcon	自定义关闭图标	ReactNode	undefined	4.9.0
content	内容	ReactNode	-	
footer	底部内容，当不需要默认底部按钮时，可以设为 `footer: null`	ReactNode | (originNode: ReactNode, extra: { OkBtn: React.FC, CancelBtn: React.FC }) => ReactNode	-	renderFu 5.9.0
getContainer	指定 Modal 挂载的 HTML 节点，false 为挂载在当前 dom	HTMLElement | () => HTMLElement | Selectors | false	document.body	
icon	自定义图标	ReactNode	<ExclamationCircleFilled />	
keyboard	是否支持键盘 esc 关闭	boolean	true	
mask	是否展示遮罩	boolean	true	
maskClosable	点击蒙层是否允许关闭	boolean	false	
okButtonProps	ok 按钮 props	ButtonProps	-	
okText	确认按钮文字	string	确定	
okType	确认按钮类型	string	`primary`	
style	可用于设置浮层的样式，调整浮层位置等	CSSProperties	-	
title	标题	ReactNode	-	
width	宽度	string | number	416	
wrapClassName	对话框外层容器的类名	string	-	4.18.0

zIndex	设置 Modal 的 z-index	number	1000	
onCancel	点击取消回调，参数为关闭函数，若返回 promise 时 resolve 为正常关闭, reject 为不关闭	function(close)	-	
onOk	点击确定回调，参数为关闭函数，若返回 promise 时 resolve 为正常关闭, reject 为不关闭	function(close)	-	

以上函数调用后，会返回一个引用，可以通过该引用更新和关闭弹窗。

```
const modal = Modal.info();

modal.update({
  title: '修改的标题',
  content: '修改的内容',
});

// 在 4.8.0 或更高版本中，可以通过传入函数的方式更新弹窗
modal.update((prevConfig) => ({
  ...prevConfig,
  title: `${prevConfig.title}（新）`,
}));

modal.destroy();
```

- `Modal.destroyAll`

使用 `Modal.destroyAll()` 可以销毁弹出的确认窗（即上述的 `Modal.info` 、 `Modal.success` 、 `Modal.error` 、 `Modal.warning` 、 `Modal.confirm` ）。通常用于路由监听当中，处理路由前进、后退不能销毁确认对话框的问题，而不用各处去使用实例的返回值进行关闭（ `modal.destroy()` 适用于主动关闭，而不是路由这样被动关闭）

```
import { browserHistory } from 'react-router';

// router change
browserHistory.listen(() => {
```

```
  Modal.destroyAll();
});
```

Modal.useModal()

当你需要使用 Context 时，可以通过 `Modal.useModal` 创建一个 `contextHolder` 插入子节点中。通过 hooks 创建的临时 Modal 将会得到 `contextHolder` 所在位置的所有上下文。创建的 `modal` 对象拥有与 `Modal.method` 相同的创建通知方法。

```
const [modal, contextHolder] = Modal.useModal();

React.useEffect(() => {
  modal.confirm({
    // ...
  });
}, []);

return <div>{contextHolder}</div>;
```

`modal.confirm` 返回方法：

- `destroy`：销毁当前窗口
- `update`：更新当前窗口
- `then`：Promise 链式调用，支持 `await` 操作。该方法为 Hooks 仅有

```
//点击 `onOk` 时返回 `true`，点击 `onCancel` 时返回 `false`
const confirmed = await modal.confirm({ ... });
```

Semantic DOM

演示

```
import React from 'react';
import type { ModalProps } from 'antd';
import { Modal } from 'antd';

import SemanticPreview from '../../../.dumi/components/SemanticPreview';
import useLocale from '../../../.dumi/hooks/useLocale';

const locales = {
  cn: {
    mask: '遮罩层元素',
    wrapper: '包裹层元素，一般用于动画容器',
    content: 'Modal 容器元素',
    header: '头部元素',
```

```
    body: '内容元素',
    footer: '底部元素',
  },
  en: {
    mask: 'Mask element',
    wrapper: 'Wrapper element. Used for motion container',
    content: 'Modal container element',
    header: 'Header element',
    body: 'Body element',
    footer: 'Footer element',
  },
};

const BlockModal = (props: ModalProps) => {
  const divRef = React.useRef<HTMLDivElement>(null);

  return (
    <div ref={divRef} style={{ position: 'absolute', inset: 0 }}>
      <Modal
        getContainer={() => divRef.current!}
        {...props}
        styles={{
          mask: {
            position: 'absolute',
            zIndex: 1,
          },
          wrapper: {
            position: 'absolute',
            zIndex: 1,
          },
        }}
        style={{
          top: '50%',
          transform: 'translateY(-50%)',
          marginBottom: 0,
          paddingBottom: 0,
        }}
      />
    </div>
  );
};

const App: React.FC = () => {
  const [locale] = useLocale(locales);
  return (
    <SemanticPreview
```

```
      componentName="Modal"
      semantics={[
        { name: 'mask', desc: locale.mask, version: '5.13.0' },
        { name: 'content', desc: locale.content, version: '5.13.0' },
        { name: 'wrapper', desc: locale.wrapper, version: '5.13.0' },
        { name: 'header', desc: locale.header, version: '5.13.0' },
        { name: 'body', desc: locale.body, version: '5.13.0' },
        { name: 'footer', desc: locale.footer, version: '5.13.0' },
      ]}
    >
      <BlockModal title="Title" closable={false} open getContainer={false}
width={400}>
        <p>Some contents...</p>
      </BlockModal>
    </SemanticPreview>
  );
};

export default App;
```

主题变量（Design Token）

FAQ

为什么 Modal 关闭时，内容不会更新?

Modal 在关闭时会将内容进行 memo 从而避免关闭过程中的内容跳跃。也因此如果你在配合使用 Form 有关闭时重置 `initialValues` 的操作，请通过在 effect 中调用 `resetFields` 来重置。

为什么 Modal 方法不能获取 context、redux、的内容和 ConfigProvider `locale/prefixCls/theme` 等配置?

直接调用 Modal 方法，antd 会通过 `ReactDOM.render` 动态创建新的 React 实体。其 context 与当前代码所在 context 并不相同，因而无法获取 context 信息。

当你需要 context 信息（例如 ConfigProvider 配置的内容）时，可以通过 `Modal.useModal` 方法会返回 `modal` 实体以及 `contextHolder` 节点。将其插入到你需要获取 context 位置即可：

```
const [modal, contextHolder] = Modal.useModal();

return (
  <Context1.Provider value="Ant">
    {/* contextHolder 在 Context1 内，它可以获得 Context1 的 context */}
    {contextHolder}
    <Context2.Provider value="Design">
      {/* contextHolder 在 Context2 外，因而不会获得 Context2 的 context */}
    </Context2.Provider>
```

```
  </Context1.Provider>
);
```

异同：通过 hooks 创建的 `contextHolder` 必须插入到子元素节点中才会生效，当你不需要上下文信息时请直接调用。

> *可通过 App 包裹组件 简化 `useModal` 等方法需要手动植入 contextHolder 的问题。*

静态方法如何设置 prefixCls？

你可以通过 `ConfigProvider.config` 进行设置。