模态对话框。

何时使用

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 `Modal.confirm()` 等语法糖方法。

API

参数	说明	类型	默认值	版本
afterClose	Modal 完全关闭后的回调	function	-	
bodyStyle	Modal body 样式	CSSProperties		
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	true	
closeIcon	自定义关闭图标	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	(确定取消按钮)	
forceRender	强制渲染 Modal	boolean	false	
getContainer	指定 Modal 挂载的 HTML 节点, false 为挂载在当前 dom	HTMLElement | () => HTMLElement | Selectors | false	document.body	
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	
maskStyle	遮罩样式	CSSProperties		
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	-	
title	标题	ReactNode	-	
visible	对话框是否可见	boolean	-	
width	宽度	string | number	520	
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)	-	

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`	
bodyStyle	Modal body 样式	CSSProperties		4.8.0
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	-	
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	<QuestionCircle />	
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	
maskStyle	遮罩样式	object	{}	
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	取消回调，参数为关闭函数，返回 promise 时 resolve 后自动关闭	function(close)	-	
onOk	点击确定回调，参数为关闭函数，返回 promise 时 resolve 后自动关闭	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>;
```

FAQ

为什么 Modal 方法不能获取 context、redux、的内容和 ConfigProvider `locale/prefixCls` 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。

如何关闭 Modal 动画?

你可以通过 `transitionName=""` 和 `maskTransitionName=""` 去除动画 CSS，但是需要注意的是。该方法为内部方法，我们不保证下个大版本重构时该属性会被保留。

静态方法如何设置 prefixCls ?

你可以通过 `ConfigProvider.config` 进行设置。