When To Use

A simple and compact dialog used for asking for user confirmation.

The difference with the `confirm` modal dialog is that it's more lightweight than the static popped full-screen confirm modal.

Examples

Basic

```tsx
import React from 'react';
import type { PopconfirmProps } from 'antd';
import { Button, message, Popconfirm } from 'antd';

const confirm: PopconfirmProps['onConfirm'] = (e) => {
  console.log(e);
  message.success('Click on Yes');
};

const cancel: PopconfirmProps['onCancel'] = (e) => {
  console.log(e);
  message.error('Click on No');
};

const App: React.FC = () => (
  <Popconfirm
    title="Delete the task"
    description="Are you sure to delete this task?"
    onConfirm={confirm}
    onCancel={cancel}
    okText="Yes"
    cancelText="No"
  >
    <Button danger>Delete</Button>
  </Popconfirm>
);

export default App;
```

Locale text

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

const App: React.FC = () => (
  <Popconfirm
    title="Delete the task"
    description="Are you sure to delete this task?"
    okText="Yes"
    cancelText="No"
  >
    <Button danger>Delete</Button>
  </Popconfirm>
);

export default App;
```

Placement

```
import React from 'react';
import { Button, ConfigProvider, Flex, Popconfirm } from 'antd';

const text = 'Are you sure to delete this task?';
const description = 'Delete the task';
const buttonWidth = 80;

const App: React.FC = () => (
  <ConfigProvider button={{ style: { width: buttonWidth, margin: 4 } }}>
    <Flex vertical justify="center" align="center" className="demo">
      <Flex justify="center" align="center" style={{ whiteSpace: 'nowrap'
}}>
        <Popconfirm
          placement="topLeft"
          title={text}
          description={description}
          okText="Yes"
          cancelText="No"
        >
          <Button>TL</Button>
        </Popconfirm>
        <Popconfirm
          placement="top"
          title={text}
          description={description}
          okText="Yes"
          cancelText="No"
        >
          <Button>Top</Button>
        </Popconfirm>
        <Popconfirm
          placement="topRight"
          title={text}
          description={description}
          okText="Yes"
          cancelText="No"
        >
          <Button>TR</Button>
        </Popconfirm>
      </Flex>
      <Flex style={{ width: buttonWidth * 5 + 32 }} justify="space-between" align="center">
        <Flex align="center" vertical>
          <Popconfirm
            placement="leftTop"
            title={text}
            description={description}
            okText="Yes"
            cancelText="No"
          >
            <Button>LT</Button>
          </Popconfirm>
          <Popconfirm
            placement="left"
            title={text}
            description={description}
            okText="Yes"
            cancelText="No"
          >
            <Button>Left</Button>
          </Popconfirm>
          <Popconfirm
            placement="leftBottom"
            title={text}
            description={description}
            okText="Yes"
            cancelText="No"
          >
            <Button>LB</Button>
          </Popconfirm>
        </Flex>
        <Flex align="center" vertical>
          <Popconfirm
            placement="rightTop"
            title={text}
            description={description}
            okText="Yes"
            cancelText="No"
          >
            <Button>RT</Button>
          </Popconfirm>
          <Popconfirm
            placement="right"
            title={text}
            description={description}
            okText="Yes"
            cancelText="No"
          >
            <Button>Right</Button>
          </Popconfirm>
          <Popconfirm
            placement="rightBottom"
            title={text}
            description={description}
            okText="Yes"
            cancelText="No"
          >
            <Button>RB</Button>
          </Popconfirm>
        </Flex>
      </Flex>
      <Flex justify="center" align="center" style={{ whiteSpace: 'nowrap' }}>
        <Popconfirm
          placement="bottomLeft"
          title={text}
          description={description}
          okText="Yes"
          cancelText="No"
        >
          <Button>BL</Button>
        </Popconfirm>
        <Popconfirm
          placement="bottom"
          title={text}
          description={description}
          okText="Yes"
          cancelText="No"
        >
          <Button>Bottom</Button>
        </Popconfirm>
        <Popconfirm
          placement="bottomRight"
          title={text}
          description={description}
          okText="Yes"
          cancelText="No"
        >
          <Button>BR</Button>
        </Popconfirm>
      </Flex>
    </Flex>
  </ConfigProvider>
);

export default App;
```

Auto Shift

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

const style: React.CSSProperties = {
  width: '300vw',
  height: '300vh',
  display: 'flex',
  alignItems: 'center',
  justifyContent: 'center',
};

const App: React.FC = () => {
  React.useEffect(() => {
    document.documentElement.scrollTop =
document.documentElement.clientHeight;
    document.documentElement.scrollLeft =
document.documentElement.clientWidth;
  }, []);
  return (
    <div style={style}>
      <Popconfirm title="Thanks for using antd. Have a nice day !" open>
        <Button type="primary">Scroll The Window</Button>
      </Popconfirm>
    </div>
  );
};

export default App;
```

Conditional trigger

```
import React, { useState } from 'react';
import { Button, message, Popconfirm, Switch } from 'antd';

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

  const changeCondition = (checked: boolean) => {
    setCondition(checked);
  };

  const confirm = () => {
    setOpen(false);
    message.success('Next step.');
  };

  const cancel = () => {
    setOpen(false);
    message.error('Click on cancel.');
  };

  const handleOpenChange = (newOpen: boolean) => {
    if (!newOpen) {
      setOpen(newOpen);
      return;
    }
    // Determining condition before show the popconfirm.
    console.log(condition);
    if (condition) {
      confirm(); // next step
    } else {
      setOpen(newOpen);
    }
  };

  return (
    <div>
      <Popconfirm
        title="Delete the task"
        description="Are you sure to delete this task?"
        open={open}
        onOpenChange={handleOpenChange}
        onConfirm={confirm}
        onCancel={cancel}
        okText="Yes"
        cancelText="No"
      >
        <Button danger>Delete a task</Button>
      </Popconfirm>
      <br />
      <br />
      Whether directly execute:
      <Switch defaultChecked onChange={changeCondition} />
    </div>
  );
};

export default App;
```

Customize icon

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

const App: React.FC = () => (
  <Popconfirm
    title="Delete the task"
    description="Are you sure to delete this task?"
    icon={<QuestionCircleOutlined style={{ color: 'red' }} />}
  >
    <Button danger>Delete</Button>
  </Popconfirm>
);

export default App;
```

Asynchronously close

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

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

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

  const handleOk = () => {
    setConfirmLoading(true);

    setTimeout(() => {
      setOpen(false);
      setConfirmLoading(false);
    }, 2000);
  };

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

  return (
    <Popconfirm
      title="Title"
      description="Open Popconfirm with async logic"
      open={open}
      onConfirm={handleOk}
      okButtonProps={{ loading: confirmLoading }}
      onCancel={handleCancel}
    >
      <Button type="primary" onClick={showPopconfirm}>
        Open Popconfirm with async logic
      </Button>
    </Popconfirm>
  );
};

export default App;
```

Asynchronously close on Promise

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

const App: React.FC = () => {
  const confirm = () =>
    new Promise((resolve) => {
      setTimeout(() => resolve(null), 3000);
    });

  return (
    <Popconfirm
      title="Title"
      description="Open Popconfirm with Promise"
      onConfirm={confirm}
      onOpenChange={() => console.log('open change')}
    >
      <Button type="primary">Open Popconfirm with Promise</Button>
    </Popconfirm>
  );
};

export default App;
```

_InternalPanelDoNotUseOrYouWillBeFired

Debug

```
import React from 'react';
import { Popconfirm } from 'antd';

const { _InternalPanelDoNotUseOrYouWillBeFired: InternalPopconfirm } =
Popconfirm;

const App: React.FC = () => (
  <>
    <InternalPopconfirm title="Are you OK?" description="Does this look
good?" />
    <InternalPopconfirm
      title="Are you OK?"
      description="Does this look good?"
      placement="bottomRight"
      style={{ width: 250 }}
    />
    <InternalPopconfirm icon={null} title="Are you OK?" />
    <InternalPopconfirm icon={null} title="Are you OK?" description="Does
this look good?" />
  </>
);

export default App;
```

Wireframe

Debug

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

const { _InternalPanelDoNotUseOrYouWillBeFired: InternalPopconfirm } =
Popconfirm;

const App: React.FC = () => (
  <ConfigProvider theme={{ token: { wireframe: true } }}>
    <InternalPopconfirm title="Are you OK?" />
    <InternalPopconfirm title="Are you OK?" placement="bottomRight" style=
{{ width: 250 }} />
  </ConfigProvider>
);

export default App;
```

API

Common props ref：[Common props]()

Param	Description	Type	Default value	Version
cancelButtonProps	The cancel button props	[ButtonProps]()	-	
cancelText	The text of the Cancel button	string	`Cancel`	
disabled	Whether show popconfirm when click its childrenNode	boolean	false	
icon	Customize icon of confirmation	ReactNode	<ExclamationCircle />	
okButtonProps	The ok button props	[ButtonProps]()	-	
okText	The text of the Confirm button	string	`OK`	
okType	Button `type` of the Confirm button	string	`primary`	
showCancel	Show cancel button	boolean	true	4.18.0
title	The title of the confirmation box	ReactNode | () => ReactNode	-	

description	The description of the confirmation box title	ReactNode | () => ReactNode	-	5.1.0
onCancel	A callback of cancel	function(e)	-	
onConfirm	A callback of confirmation	function(e)	-	
onPopupClick	A callback of popup click	function(e)	-	5.5.0

404

The requested path could not be found

Semantic DOM

演示

```
import React from 'react';
import { Popconfirm } from 'antd';

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

const locales = {
  cn: {
    root: '根元素（包含箭头、内容元素）',
    body: '内容元素',
  },
  en: {
    root: 'Root element (including arrows, content elements)',
    body: 'Body element',
  },
};

const BlockList: React.FC<React.PropsWithChildren> = (props: any) => {
  const divRef = React.useRef<HTMLDivElement>(null);

  return (
    <div ref={divRef} style={{ position: 'absolute', marginTop: 60 }}>
      <Popconfirm
        title="prompt text"
        open
        placement="top"
        autoAdjustOverflow={false}
        getPopupContainer={() => divRef.current}
        {...props}
      />
    </div>
  );
};

const App: React.FC = () => {
  const [locale] = useLocale(locales);
  return (
    <SemanticPreview
      componentName="Popconfirm"
      semantics={[
        { name: 'root', desc: locale.root, version: '5.23.0' },
        { name: 'body', desc: locale.body, version: '5.23.0' },
      ]}
    >
      <BlockList />
    </SemanticPreview>
  );
};

export default App;
```

Design Token

FAQ

404

The requested path could not be found

For more questions, please refer to Tooltip FAQ.