## 何时使用 {#when-to-use}

当目标元素有进一步的描述和相关操作时，可以收纳到卡片中，根据用户的操作行为进行展现。

和 `Tooltip` 的区别是，用户可以对浮层上的元素进行操作，因此它可以承载更复杂的内容，比如链接或按钮等。

## 代码演示

### 基本

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

const content = (
  <div>
    <p>Content</p>
    <p>Content</p>
  </div>
);

const App: React.FC = () => (
  <Popover content={content} title="Title">
    <Button type="primary">Hover me</Button>
  </Popover>
);

export default App;
```

### 三种触发方式

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

const content = (
  <div>
    <p>Content</p>
    <p>Content</p>
  </div>
);

const App: React.FC = () => (
  <Space wrap>
    <Popover content={content} title="Title" trigger="hover">
      <Button>Hover me</Button>
    </Popover>
    <Popover content={content} title="Title" trigger="focus">
      <Button>Focus me</Button>
    </Popover>
    <Popover content={content} title="Title" trigger="click">
      <Button>Click me</Button>
    </Popover>
  </Space>
);

export default App;
```

**位置**

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

const text = <span>Title</span>;

const content = (
  <div>
    <p>Content</p>
    <p>Content</p>
  </div>
);

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'
}}>
        <Popover placement="topLeft" title={text} content={content}>
          <Button>TL</Button>
        </Popover>
        <Popover placement="top" title={text} content={content}>
          <Button>Top</Button>
        </Popover>
        <Popover placement="topRight" title={text} content={content}>
          <Button>TR</Button>
        </Popover>
      </Flex>
      <Flex style={{ width: buttonWidth * 5 + 32 }} justify="space-between"
align="center">
        <Flex align="center" vertical>
          <Popover placement="leftTop" title={text} content={content}>
```

```
            <Button>LT</Button>
          </Popover>
          <Popover placement="left" title={text} content={content}>
            <Button>Left</Button>
          </Popover>
          <Popover placement="leftBottom" title={text} content={content}>
            <Button>LB</Button>
          </Popover>
        </Flex>
        <Flex align="center" vertical>
          <Popover placement="rightTop" title={text} content={content}>
            <Button>RT</Button>
          </Popover>
          <Popover placement="right" title={text} content={content}>
            <Button>Right</Button>
          </Popover>
          <Popover placement="rightBottom" title={text} content={content}>
            <Button>RB</Button>
          </Popover>
        </Flex>
      </Flex>
      <Flex justify="center" align="center" style={{ whiteSpace: 'nowrap'
}}>
        <Popover placement="bottomLeft" title={text} content={content}>
          <Button>BL</Button>
        </Popover>
        <Popover placement="bottom" title={text} content={content}>
          <Button>Bottom</Button>
        </Popover>
        <Popover placement="bottomRight" title={text} content={content}>
          <Button>BR</Button>
        </Popover>
      </Flex>
    </Flex>
  </ConfigProvider>
);

export default App;
```

**箭头展示**

```
import React, { useMemo, useState } from 'react';
import { Button, ConfigProvider, Flex, Popover, Segmented } from 'antd';
import type { PopoverProps } from 'antd';

const text = <span>Title</span>;
const buttonWidth = 80;

const content = (
  <div>
    <p>Content</p>
    <p>Content</p>
  </div>
);

const App: React.FC = () => {
  const [arrow, setArrow] = useState<'Show' | 'Hide' | 'Center'>('Show');

  const mergedArrow = useMemo<PopoverProps['arrow']>(() => {
    if (arrow === 'Hide') {
      return false;
    }

    if (arrow === 'Show') {
      return true;
    }

    return {
      pointAtCenter: true,
    };
  }, [arrow]);

  return (
    <ConfigProvider button={{ style: { width: buttonWidth, margin: 4 } }}>
      <Segmented
        options={['Show', 'Hide', 'Center']}
        onChange={setArrow}
        style={{ marginBottom: 24 }}
      />
      <Flex vertical justify="center" align="center" className="demo">
        <Flex justify="center" align="center" style={{ whiteSpace: 'nowrap'
}}>
          <Popover placement="topLeft" title={text} content={content}
arrow={mergedArrow}>
            <Button>TL</Button>
          </Popover>
          <Popover placement="top" title={text} content={content} arrow=
{mergedArrow}>
            <Button>Top</Button>
          </Popover>
          <Popover placement="topRight" title={text} content={content}
```

```
arrow={mergedArrow}>
            <Button>TR</Button>
          </Popover>
        </Flex>
        <Flex style={{ width: buttonWidth * 5 + 32 }} justify="space-
between" align="center">
          <Flex align="center" vertical>
            <Popover placement="leftTop" title={text} content={content}
arrow={mergedArrow}>
              <Button>LT</Button>
            </Popover>
            <Popover placement="left" title={text} content={content} arrow=
{mergedArrow}>
              <Button>Left</Button>
            </Popover>
            <Popover placement="leftBottom" title={text} content={content}
arrow={mergedArrow}>
              <Button>LB</Button>
            </Popover>
          </Flex>
          <Flex align="center" vertical>
            <Popover placement="rightTop" title={text} content={content}
arrow={mergedArrow}>
              <Button>RT</Button>
            </Popover>
            <Popover placement="right" title={text} content={content}
arrow={mergedArrow}>
              <Button>Right</Button>
            </Popover>
            <Popover placement="rightBottom" title={text} content={content}
arrow={mergedArrow}>
              <Button>RB</Button>
            </Popover>
          </Flex>
        </Flex>
        <Flex justify="center" align="center" style={{ whiteSpace: 'nowrap'
}}>
          <Popover placement="bottomLeft" title={text} content={content}
arrow={mergedArrow}>
            <Button>BL</Button>
          </Popover>
          <Popover placement="bottom" title={text} content={content} arrow=
{mergedArrow}>
            <Button>Bottom</Button>
          </Popover>
          <Popover placement="bottomRight" title={text} content={content}
```

```
arrow={mergedArrow}>
            <Button>BR</Button>
          </Popover>
        </Flex>
      </Flex>
    </ConfigProvider>
  );
};

export default App;
```

### Arrow.pointAtCenter

Debug

```
import React from 'react';
import { createStyles } from 'antd-style';
import { Flex, Popover } from 'antd';
import type { GetProp } from 'antd';

const useStyle = createStyles(() => ({
  item: {
    width: '280px',
    height: '280px',
    display: 'inline-flex',
    justifyContent: 'center',
    alignItems: 'center',
    border: '1px dashed purple',
  },

  box: {
    width: '40px',
    height: '40px',
    backgroundColor: 'deepskyblue',
  },

  cross: {
    position: 'relative',

    '&::before, &::after': {
      content: '""',
      position: 'absolute',
      inset: 0,
    },
    '&::before': {
      top: '50%',
      height: '1px',
      backgroundColor: 'red',
    },
    '&::after': {
      left: '50%',
      width: '1px',
      backgroundColor: 'blue',
    },
  },
}));

type Placement = GetProp<typeof Popover, 'placement'>;

const placements: Placement[] = [
  'topLeft',
  'top',
  'topRight',
  'leftTop',
  'left',
  'leftBottom',
  'rightTop',
  'right',
  'rightBottom',
  'bottomLeft',
  'bottom',
  'bottomRight',
];

const App = () => {
  const { styles, cx } = useStyle();
  return (
    <Flex gap={16} wrap>
      {placements.map((placement) => (
        <div key={placement} className={styles.item}>
          <Popover
            placement={placement}
            content={
              <Flex align="center" justify="center">
                {placement}
              </Flex>
            }
            autoAdjustOverflow={false}
            arrow={{ pointAtCenter: true }}
            forceRender
            open
          >
```

```
            <div className={cx(styles.box, styles.cross)} />
          </Popover>
        </div>
      ))}
    </Flex>
  );
};

export default App;
```

**贴边偏移**

```
import React from 'react';
import { Button, Popover } 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}>
      <Popover content="Thanks for using antd. Have a nice day !" open>
        <Button type="primary">Scroll The Window</Button>
      </Popover>
    </div>
  );
};

export default App;
```

**从浮层内关闭**

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

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

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

  const handleOpenChange = (newOpen: boolean) => {
    setOpen(newOpen);
  };

  return (
    <Popover
      content={<a onClick={hide}>Close</a>}
      title="Title"
      trigger="click"
      open={open}
      onOpenChange={handleOpenChange}
    >
      <Button type="primary">Click me</Button>
    </Popover>
  );
};

export default App;
```

**悬停点击弹出窗口**

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

const App: React.FC = () => {
  const [clicked, setClicked] = useState(false);
  const [hovered, setHovered] = useState(false);

  const hide = () => {
    setClicked(false);
    setHovered(false);
  };

  const handleHoverChange = (open: boolean) => {
    setHovered(open);
    setClicked(false);
  };
  const handleClickChange = (open: boolean) => {
    setHovered(false);
    setClicked(open);
  };

  const hoverContent = <div>This is hover content.</div>;
  const clickContent = <div>This is click content.</div>;
  return (
    <Popover
      style={{ width: 500 }}
      content={hoverContent}
      title="Hover title"
      trigger="hover"
      open={hovered}
      onOpenChange={handleHoverChange}
    >
      <Popover
        content={
          <div>
            {clickContent}
            <a onClick={hide}>Close</a>
          </div>
        }
        title="Click title"
        trigger="click"
        open={clicked}
        onOpenChange={handleClickChange}
      >
        <Button>Hover and click / 悬停并单击</Button>
      </Popover>
    </Popover>
  );
};

export default App;
```

### _InternalPanelDoNotUseOrYouWillBeFired

Debug

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

const { _InternalPanelDoNotUseOrYouWillBeFired: InternalPopover } =
Popover;
```

```
const content = (
  <div>
    <p>Content</p>
    <p>Content</p>
  </div>
);

const App: React.FC = () => (
  <>
    <InternalPopover content={content} title="Title" />
    <InternalPopover
      content={content}
      title="Title"
      placement="bottomLeft"
      style={{ width: 250 }}
    />
  </>
);

export default App;
```

### 线框风格

Debug

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

const { _InternalPanelDoNotUseOrYouWillBeFired: InternalPopover } =
Popover;

const content = (
  <div>
    <p>Content</p>
    <p>Content</p>
  </div>
);

const App: React.FC = () => (
  <ConfigProvider theme={{ token: { wireframe: true } }}>
    <InternalPopover content={content} title="Title" />
    <InternalPopover
      content={content}
      title="Title"
      placement="bottomLeft"
      style={{ width: 250 }}
    />
  </ConfigProvider>
);

export default App;
```

### 组件 Token

Debug

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

const { _InternalPanelDoNotUseOrYouWillBeFired: InternalPopover } =
Popover;

const content = (
  <div>
    <p>Content</p>
    <p>Content</p>
  </div>
);

const App: React.FC = () => (
  <ConfigProvider
    theme={{
      components: {
        Popover: {
          titleMinWidth: 40,
        },
      },
    }}
  >
    <InternalPopover content={content} title="Title" />
    <InternalPopover
      content={content}
      title="Title"
      placement="bottomLeft"
      style={{ width: 250 }}
    />
  </ConfigProvider>
);

export default App;
```

## API

通用属性参考：[通用属性]

| 参数 | 说明 | 类型 | 默认值 | 版本 |
|---|---|---|---|---|
| content | 卡片内容 | ReactNode \| () => ReactNode | - | |
| title | 卡片标题 | ReactNode \| () => ReactNode | - | |

404

The requested path could not be found

## 注意

请确保 `Popover` 的子元素能接受 `onMouseEnter` 、 `onMouseLeave` 、 `onFocus` 、 `onClick` 事件。

## Semantic DOM

### 演示

```
import React from 'react';
import { Popover } 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 }}>
      <Popover
        title="prompt text"
        open
        placement="top"
        autoAdjustOverflow={false}
        getPopupContainer={() => divRef.current}
        {...props}
      />
    </div>
  );
};

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

export default App;
```

## 主题变量（Design Token）

## FAQ

404

The requested path could not be found

更多问题，请参考 Tooltip FAQ。