Popover

The floating card popped by clicking or hovering.

When To Use

A simple popup menu to provide extra information or operations.

Comparing with `Tooltip`, besides information `Popover` card can also provide action elements like links and buttons.

Examples

Basic

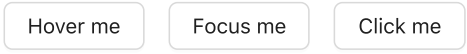
The most basic example. The size of the floating layer depends on the contents region.

```
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;
```



Three ways to trigger

Mouse to click, focus and move in.

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

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

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

export default App;
```

Placement

There are 12 `placement` options available.

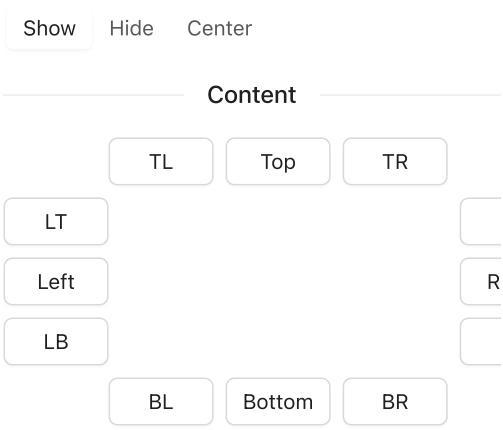
```
import { Button, Popover } from 'antd';
import React from 'react';

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

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

const buttonWidth = 70;

const App: React.FC = () => (
  <div>
    <div style={{ marginLeft: buttonWidth, whi
      <Popover placement="topLeft" title={text
        <Button>TL</Button>
      </Popover>
      <Popover placement="top" title={text} co
        <Button>Top</Button>
      </Popover>
      <Popover placement="topRight" title={te
        <Button>TR</Button>
      </Popover>
    </div>
    <div style={{ width: buttonWidth, float: '
      <Popover placement="leftTop" title={text
        <Button>LT</Button>
      </Popover>
      <Popover placement="left" title={text} c
        <Button>Left</Button>
      </Popover>
      <Popover placement="leftBottom" title={t
        <Button>LB</Button>
      </Popover>
    </div>
    <div style={{ width: buttonWidth, marginLe
      <Popover placement="rightTop" title={te
        <Button>RT</Button>
      </Popover>
      <Popover placement="right" title={text}
        <Button>Right</Button>
      </Popover>
      <Popover placement="rightBottom" title=
        <Button>RB</Button>
      </Popover>
    </div>
    <div style={{ marginLeft: buttonWidth, cle
      <Popover placement="bottomLeft" title={t
        <Button>BL</Button>
      </Popover>
      <Popover placement="bottom" title={text}
        <Button>Bottom</Button>
      </Popover>
```



Arrow

Hide arrow by `arrow`.

```
import React, { useMemo, useState } from 'reac
import { Button, Divider, Popover, Segmented }

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

const buttonWidth = 70;

const App: React.FC = () => {
  const [showArrow, setShowArrow] = useState(t
  const [arrowAtCenter, setArrowAtCenter] = us

  const mergedArrow = useMemo(() => {
    if (arrowAtCenter) return { pointAtCenter:
    return showArrow;
  }, [showArrow, arrowAtCenter]);

  return (
    <div className="demo">
      <Segmented
        options={['Show', 'Hide', 'Center']}
        onChange={(val) => {
          setShowArrow(val !== 'Hide');
          setArrowAtCenter(val === 'Center');
        }}
      />
      <Divider orientation="center">Content</D
      <div style={{ marginLeft: buttonWidth, w
        <Popover placement="topLeft" title={te
          <Button>TL</Button>
        </Popover>
        <Popover placement="top" title={text}
          <Button>Top</Button>
        </Popover>
        <Popover placement="topRight" title={t
          <Button>TR</Button>
        </Popover>
      </div>
      <div style={{ width: buttonWidth, float:
        <Popover placement="leftTop" title={te
          <Button>LT</Button>
        </Popover>
        <Popover placement="left" title={text}
          <Button>Left</Button>
        </Popover>
        <Popover placement="leftBottom" title=
```

Controlling the close of the dialog

Use `open` prop to control the display of the card.

```
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;
```

Hover with click popover

The following example shows how to create a popover which can be hovered and clicked.

```
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 cont
  const clickContent = <div>This is click cont
  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</B
      </Popover>
    </Popover>
  );
};

export default App;
```

API

Param	Description	Type	Default value	Version
content	Content of the card	ReactNode | () => ReactNode	-	
title	Title of the card	ReactNode | () => ReactNode	-	

Consult Tooltip's documentation to find more APIs.

Note

Please ensure that the child node of `Popover` accepts `onMouseEnter`, `onMouseLeave`, `onFocus`, `onClick` events.

Design Token

▼ Global Token

Token Name	Description	Type	Default Value
colorBgElevated	Container background color of the popup layer, in dark mode the color value of this token will be a little brighter than `colorBgContainer`. E.g: modal, pop-up, menu, etc.	`string`	`#ffffff`
colorSplit	Used as the color of separator, this color is the same as colorBorderSecondary but with transparency.	`string`	`rgba(5, 5, 5, 0.06)`
colorText	Default text color which comply with W3C standards, and this color is also the darkest neutral color.	`string`	`rgba(0, 0, 0, 0.88)`
colorTextHeading	Control the font color of heading.	`string`	`rgba(0, 0, 0, 0.88)`
borderRadiusLG	LG size border radius, used in some large border radius components, such as Card, Modal and other components.	`number`	8
borderRadiusOuter	Outer border radius	`number`	4
borderRadiusXS	XS size border radius, used in some small border radius components, such as Segmented, Arrow and other components with small border radius.	`number`	2
boxShadowSecondary	Control the secondary box shadow style of an element.	`string`	0 6px 16px 0 rgba(0, 0, 0, 0.08), 0 3px 6px -4px rgba(0, 0, 0, 0.12), 0 9px 28px 8px rgba(0, 0, 0, 0.05)
controlHeight	The height of the basic controls such as buttons and input boxes in Ant Design	`number`	32
fontFamily	The font family of Ant Design prioritizes the default interface font of the system, and	`string`	-apple-system, BlinkMacSystemFont, 'Segoe UI',

Token Name	Description	Type	Default Value
	provides a set of alternative font libraries that are suitable for screen display to maintain the readability and readability of the font under different platforms and browsers, reflecting the friendly, stable and professional characteristics.		Roboto, 'Helvetica Neue', Arial, 'Noto Sans', sans-serif, 'Apple Color Emoji', 'Segoe UI Emoji', 'Segoe UI Symbol', 'Noto Color Emoji'
fontSize	The most widely used font size in the design system, from which the text gradient will be derived.	`number`	14
fontWeightStrong	Control the font weight of heading components (such as h1, h2, h3) or selected item.	`number`	600
lineHeight	Line height of text.	`number`	1.5714285714285714
lineType	Border style of base components	`string`	solid
lineWidth	Border width of base components	`number`	1
marginXS	Control the margin of an element, with a small size.	`number`	8
motionDurationMid	Motion speed, medium speed. Used for medium element animation interaction.	`string`	0.2s
motionEaseInOutCirc	Preset motion curve.	`string`	cubic-bezier(0.78, 0.14, 0.15, 0.86)
motionEaseOutCirc	Preset motion curve.	`string`	cubic-bezier(0.08, 0.82, 0.17, 1)
padding	Control the padding of the element.	`number`	16
paddingSM	Control the small padding of the element.	`number`	12
sizePopupArrow	The size of the component arrow	`number`	16
wireframe	Used to change the visual effect of the component to wireframe, if you need to use the V4 effect, you need to enable the configuration item	`boolean`	false
zIndexPopupBase	Base zIndex of component like FloatButton, Affix which can be cover by large popup	`number`	1000