何时使用 {#when-to-use}

鼠标移入则显示提示，移出消失，气泡浮层不承载复杂文本和操作。

可用来代替系统默认的 `title` 提示，提供一个 `按钮/文字/操作` 的文案解释。

代码演示

基本

```tsx
import React from 'react';
import { Tooltip } from 'antd';

const App: React.FC = () => (
  <Tooltip title="prompt text">
    <span>Tooltip will show on mouse enter.</span>
  </Tooltip>
);

export default App;
```

位置

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

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

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

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

export default App;
```

箭头展示

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

```tsx
import type { TooltipProps } from 'antd';

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

const buttonWidth = 80;

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

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

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

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

  return (
    <ConfigProvider button={{ style: { width: buttonWidth, margin: 4 } }}>
      <Segmented
        value={arrow}
        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'
}}>
          <Tooltip placement="topLeft" title={text} arrow={mergedArrow}>
            <Button>TL</Button>
          </Tooltip>
          <Tooltip placement="top" title={text} arrow={mergedArrow}>
            <Button>Top</Button>
          </Tooltip>
          <Tooltip placement="topRight" title={text} arrow={mergedArrow}>
            <Button>TR</Button>
          </Tooltip>
        </Flex>
        <Flex style={{ width: buttonWidth * 5 + 32 }} justify="space-
between" align="center">
```

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

export default App;
```

贴边偏移

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

export default App;
```

自动调整位置

Debug

```tsx
import React from 'react';
import type { TooltipProps } from 'antd';
import { Button, Tooltip, Typography } from 'antd';

const Block = React.forwardRef<HTMLDivElement, Partial<TooltipProps>>
((props, ref) => (
  <div
    style={{
      overflow: 'auto',
      position: 'relative',
      padding: '24px',
      border: '1px solid #e9e9e9',
    }}
```

```
    ref={ref}
  >
    <div
      style={{
        width: '200%',
        display: 'flex',
        flexDirection: 'column',
        alignItems: 'center',
        rowGap: 16,
      }}
    >
      <Tooltip {...props} placement="left" title="Prompt Text">
        <Button>Adjust automatically / 自动调整</Button>
      </Tooltip>
      <Tooltip {...props} placement="left" title="Prompt Text"
autoAdjustOverflow={false}>
        <Button>Ignore / 不处理</Button>
      </Tooltip>
    </div>
  </div>
));

const App: React.FC = () => {
  const containerRef1 = React.useRef<HTMLDivElement>(null);
  const containerRef2 = React.useRef<HTMLDivElement>(null);

  React.useEffect(() => {
    containerRef1.current!.scrollLeft = containerRef1.current!.clientWidth
* 0.5;
    containerRef2.current!.scrollLeft = containerRef2.current!.clientWidth
* 0.5;
  }, []);

  return (
    <div style={{ display: 'flex', flexDirection: 'column', rowGap: 16 }}>
      <Typography.Title level={5}>With
`getPopupContainer`</Typography.Title>
      <Block ref={containerRef1} getPopupContainer={(trigger) =>
trigger.parentElement!} />

      <Typography.Title level={5}>Without
`getPopupContainer`</Typography.Title>
      <Block ref={containerRef2} />
    </div>
  );
};
```

```
export default App;
```

隐藏后销毁

Debug

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

const App: React.FC = () => (
  <Tooltip destroyTooltipOnHide title="prompt text">
    <span>Tooltip will destroy when hidden.</span>
  </Tooltip>
);

export default App;
```

多彩文字提示

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

const colors = [
  'pink',
  'red',
  'yellow',
  'orange',
  'cyan',
  'green',
  'blue',
  'purple',
  'geekblue',
  'magenta',
  'volcano',
  'gold',
  'lime',
];

const customColors = ['#f50', '#2db7f5', '#87d068', '#108ee9'];

const App: React.FC = () => (
  <>
    <Divider orientation="left">Presets</Divider>
    <Space wrap>
      {colors.map((color) => (
```

```
        <Tooltip title="prompt text" color={color} key={color}>
          <Button>{color}</Button>
        </Tooltip>
      ))}
    </Space>
    <Divider orientation="left">Custom</Divider>
    <Space wrap>
      {customColors.map((color) => (
        <Tooltip title="prompt text" color={color} key={color}>
          <Button>{color}</Button>
        </Tooltip>
      ))}
    </Space>
  </>
);

export default App;
```

_InternalPanelDoNotUseOrYouWillBeFired

Debug

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

const { _InternalPanelDoNotUseOrYouWillBeFired: InternalTooltip } =
Tooltip;

const App: React.FC = () => (
  <>
    <InternalTooltip title="Hello, Pink Pure Panel!" color="pink" />
    <InternalTooltip title="Hello, Customize Color Pure Panel!"
color="#f50" />
    <InternalTooltip title="Hello, Pure Panel!" placement="bottomLeft"
style={{ width: 200 }} />
  </>
);

export default App;
```

Debug

Debug

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

```tsx
const zeroWidthEle = <div />;

const App: React.FC = () => (
  <Flex vertical gap={72} align="flex-start">
    <span />
    <Tooltip
      open
      title="Thanks for using antd. Have a nice day !"
      arrow={{ pointAtCenter: true }}
      placement="topLeft"
    >
      <Button>Point at center</Button>
    </Tooltip>
    <Tooltip open title={zeroWidthEle} placement="topLeft">
      <Button>Min Width</Button>
    </Tooltip>
    <Tooltip open title={zeroWidthEle} placement="top">
      <Button>Min Width</Button>
    </Tooltip>
  </Flex>
);

export default App;
```

禁用

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

const App: React.FC = () => {
  const [disabled, setDisabled] = useState(true);

  return (
    <Tooltip title={disabled ? null : 'prompt text'}>
      <Button onClick={() => setDisabled(!disabled)}>{disabled ? 'Enable' :
'Disable'}</Button>
    </Tooltip>
  );
};

export default App;
```

禁用子元素

Debug

```
import React from 'react';
import { Button, Checkbox, Input, InputNumber, Select, Space, Tooltip }
from 'antd';

const WrapperTooltip: React.FC<React.PropsWithChildren> = (props) => (
  <Tooltip title="Thanks for using antd. Have a nice day !" {...props} />
);

const App: React.FC = () => (
  <Space>
    <WrapperTooltip>
      <Button disabled>Disabled</Button>
    </WrapperTooltip>
    <WrapperTooltip>
      <Input disabled placeholder="disabled" />
    </WrapperTooltip>
    <WrapperTooltip>
      <InputNumber disabled />
    </WrapperTooltip>
    <WrapperTooltip>
      <Checkbox disabled />
    </WrapperTooltip>
    <WrapperTooltip>
      <Select disabled />
    </WrapperTooltip>
  </Space>
);

export default App;
```

自定义子组件

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

const ComponentWithEvents: React.FC<React.HTMLAttributes<HTMLSpanElement>>
= (props) => (
  <span {...props}>This text is inside a component with the necessary
events exposed.</span>
);

const App: React.FC = () => (
  <Tooltip title="prompt text">
    <ComponentWithEvents />
  </Tooltip>
```

```
);

export default App;
```

API

通用属性参考：通用属性

参数	说明	类型	默认值
title	提示文字	ReactNode | () => ReactNode	-

共同的 API

404

The requested path could not be found

Semantic DOM

演示

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

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

为何 Tooltip 的内容在关闭时不会更新?

Tooltip 默认在关闭时会缓存内容，以防止内容更新时出现闪烁：

```
// `title` 不会因为 `user` 置空而闪烁置空
<Tooltip open={user} title={user?.name} />
```

如果需要在关闭时也更新内容，可以设置 `fresh` 属性（例如 #44830 中的场景）：

```
<Tooltip open={user} title={user?.name} fresh />
```

404

The requested path could not be found