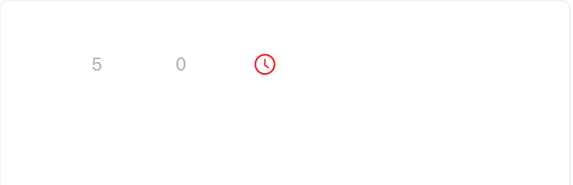
Badge

Small numerical value or status descriptor for UI elements.

When To Use

Badge normally appears in proximity to notifications or user avatars with eye-catching appeal, typically displaying unread messages count.

Examples



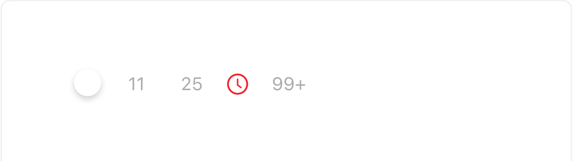
Basic

Simplest Usage. Badge will be hidden when `count` is `0`, but we can use `showZero` to show it.

```
import React from 'react';
import { ClockCircleOutlined } from '@ant-desi
import { Avatar, Badge, Space } from 'antd';

const App: React.FC = () => (
  <Space size="middle">
    <Badge count={5}>
      <Avatar shape="square" size="large" />
    </Badge>
    <Badge count={0} showZero>
      <Avatar shape="square" size="large" />
    </Badge>
    <Badge count={<ClockCircleOutlined style={
      <Avatar shape="square" size="large" />
    </Badge>
  </Space>
);

export default App;
```



Standalone

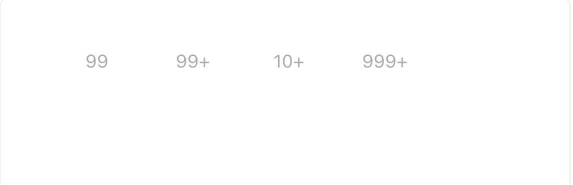
Used in standalone when children is empty.

```
import React, { useState } from 'react';
import { ClockCircleOutlined } from '@ant-desi
import { Badge, Space, Switch } from 'antd';

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

  return (
    <Space>
      <Switch checked={show} onChange={() => s
      <Badge count={show ? 11 : 0} showZero cc
      <Badge count={show ? 25 : 0} />
      <Badge count={show ? <ClockCircleOutline
      <Badge
        className="site-badge-count-109"
        count={show ? 109 : 0}
        style={{ backgroundColor: '#52c41a' }}
      />
    </Space>
  );
};

export default App;
```



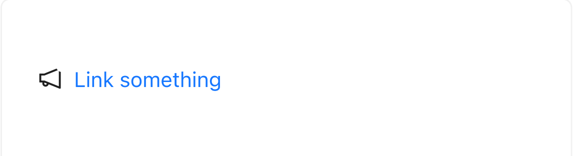
Overflow Count

`${overflowCount}+` is displayed when count is larger than `overflowCount`. The default value of `overflowCount` is `99`.

```
import React from 'react';
import { Avatar, Badge, Space } from 'antd';

const App: React.FC = () => (
  <Space size="large">
    <Badge count={99}>
      <Avatar shape="square" size="large" />
    </Badge>
    <Badge count={100}>
      <Avatar shape="square" size="large" />
    </Badge>
    <Badge count={99} overflowCount={10}>
      <Avatar shape="square" size="large" />
    </Badge>
    <Badge count={1000} overflowCount={999}>
      <Avatar shape="square" size="large" />
    </Badge>
  </Space>
);

export default App;
```



Red badge

This will simply display a red badge, without a specific count. If count equals 0, it won't display the dot.

```
import React from 'react';
import { NotificationOutlined } from '@ant-des
import { Badge, Space } from 'antd';

const App: React.FC = () => (
  <Space>
    <Badge dot>
      <NotificationOutlined style={{ fontSize:
    </Badge>
    <Badge dot>
      <a href="#">Link something</a>
    </Badge>
  </Space>
);

export default App;
```

5

− + ?

Dynamic

The count will be animated as it changes.

```
import React, { useState } from 'react';
import { MinusOutlined, PlusOutlined, Questior
import { Avatar, Badge, Button, Switch, Space

const ButtonGroup = Button.Group;

const App: React.FC = () => {
  const [count, setCount] = useState(5);
  const [show, setShow] = useState(true);

  const increase = () => {
    setCount(count + 1);
  };

  const decline = () => {
    let newCount = count - 1;
    if (newCount < 0) {
      newCount = 0;
    }
    setCount(newCount);
  };

  const random = () => {
    const newCount = Math.floor(Math.random()
    setCount(newCount);
  };

  const onChange = (checked: boolean) => {
    setShow(checked);
  };

  return (
    <Space direction="vertical">
      <Space size="large">
        <Badge count={count}>
          <Avatar shape="square" size="large"
        </Badge>
        <ButtonGroup>
          <Button onClick={decline} icon={<Mir
          <Button onClick={increase} icon={<Pl
          <Button onClick={random} icon={<Ques
        </ButtonGroup>
      </Space>
      <Space size="large">
        <Badge dot={show}>
          <Avatar shape="square" size="large"
        </Badge>
        <Switch onChange={onChange} checked={s
      </Space>
    </Space>
  );
};

export default App;
```

5

Clickable

The badge can be wrapped with `a` tag to make it linkable.

```
import React from 'react';
import { Avatar, Badge } from 'antd';

const App: React.FC = () => (
  <a href="#">
    <Badge count={5}>
      <Avatar shape="square" size="large" />
    </Badge>
  </a>
);

export default App;
```

5 5

Size

Set size of numeral Badge.

```
import React from 'react';
import { Avatar, Badge, Space } from 'antd';

const App: React.FC = () => (
  <Space size="middle">
    <Badge size="default" count={5}>
      <Avatar shape="square" size="large" />
    </Badge>
    <Badge size="small" count={5}>
      <Avatar shape="square" size="large" />
    </Badge>
  </Space>
);

export default App;
```

5

Offset

Set offset of the badge dot, the format is `[left, top]`, which represents the offset of the status dot from the left and top of the default position.

```
import React from 'react';
import { Avatar, Badge } from 'antd';

const App: React.FC = () => (
  <Badge count={5} offset={[10, 10]}>
    <Avatar shape="square" size="large" />
  </Badge>
);

export default App;
```

Success

Error

Default

Processing

Warning

Status

Standalone badge with status.

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

const App: React.FC = () => (
  <>
    <Space>
      <Badge status="success" />
      <Badge status="error" />
      <Badge status="default" />
      <Badge status="processing" />
      <Badge status="warning" />
    </Space>
    <br />
    <Space direction="vertical">
      <Badge status="success" text="Success" /
      <Badge status="error" text="Error" />
      <Badge status="default" text="Default" /
      <Badge status="processing" text="Process
      <Badge status="warning" text="Warning" /
    </Space>
  </>
);

export default App;
```

Presets

pink

red

yellow

orange

cyan

green

blue

purple

geekblue

magenta

volcano

gold

lime

Custom

#f50

rgb(45, 183, 245)

hsl(102, 53%, 61%)

hwb(205 6% 9%)

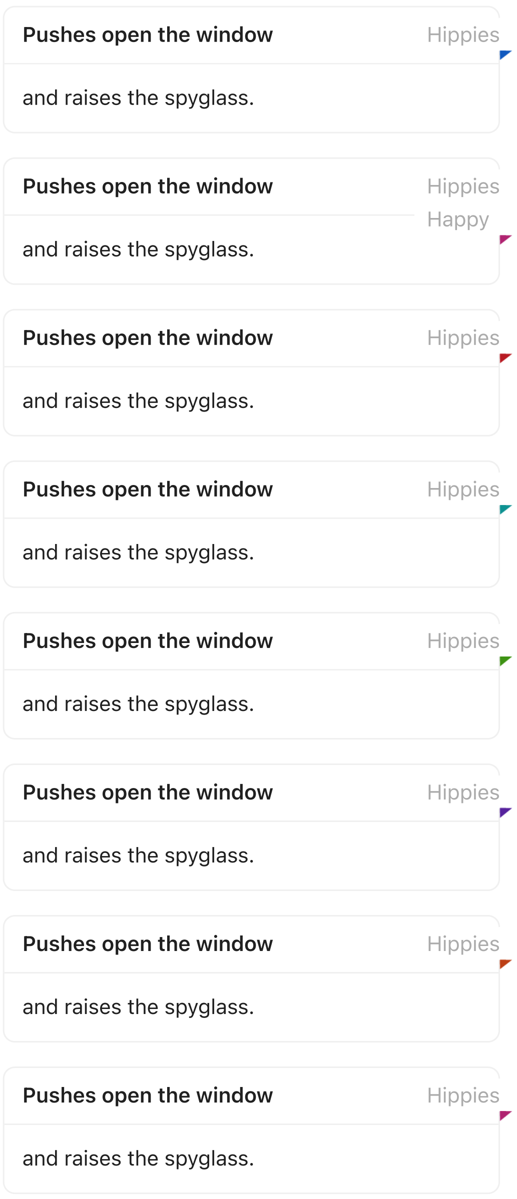
Colorful Badge

We preset a series of colorful Badge styles for use in different situations. You can also set it to a hex color string for custom color.

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

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

const App: React.FC = () => (
  <>
    <Divider orientation="left">Presets</Divic
    <Space direction="vertical">
      {colors.map((color) => (
        <Badge key={color} color={color} text=
      ))}
    </Space>
    <Divider orientation="left">Custom</Divide
    <Space direction="vertical">
      <Badge color="#f50" text="#f50" />
      <Badge color="rgb(45, 183, 245)" text="r
      <Badge color="hsl(102, 53%, 61%)" text='
```


Pushes open the window Hippies

and raises the spyglass.

Pushes open the window Hippies
Happy

and raises the spyglass.

Pushes open the window Hippies

and raises the spyglass.

Pushes open the window Hippies

and raises the spyglass.

Pushes open the window Hippies

and raises the spyglass.

Pushes open the window Hippies

and raises the spyglass.

Pushes open the window Hippies

and raises the spyglass.

Pushes open the window Hippies

and raises the spyglass.

Ribbon

Use ribbon badge.

```
import React from 'react';
import { Badge, Card, Space } from 'antd';

const App: React.FC = () => (
  <Space direction="vertical" size="middle" st
    <Badge.Ribbon text="Hippies">
      <Card title="Pushes open the window" siz
        and raises the spyglass.
      </Card>
    </Badge.Ribbon>
    <Badge.Ribbon text={
        <div>
          Hippies <br />
          Happy
        </div>
      } color="pink">
      <Card title="Pushes open the window" siz
        and raises the spyglass.
      </Card>
    </Badge.Ribbon>
    <Badge.Ribbon text="Hippies" color="red">
      <Card title="Pushes open the window" siz
        and raises the spyglass.
      </Card>
    </Badge.Ribbon>
```

```
      <Badge color="hsl(102, 53%, 61%)" text=
      <Badge color="hwb(205 6% 9%)" text="hwb(
    </Space>
  </>
);

export default App;
```

```
    <Badge.Ribbon text="Hippies" color="cyan">
      <Card title="Pushes open the window" siz
        and raises the spyglass.
      </Card>
    </Badge.Ribbon>
    <Badge.Ribbon text="Hippies" color="green'
      <Card title="Pushes open the window" siz
        and raises the spyglass.
      <Card title="Pushes open the window" siz
        and raises the spyglass.
      </Card>
    </Badge.Ribbon>
    <Badge.Ribbon text="Hippies" color="volcar
      <Card title="Pushes open the window" siz
        and raises the spyglass.
      </Card>
    </Badge.Ribbon>
    <Badge.Ribbon text="Hippies" color="magent
      <Card title="Pushes open the window" siz
        and raises the spyglass.
      </Card>
    </Badge.Ribbon>
  </Space>
);

export default App;
```

Property	Description	Type	Default	Version
color	Customize Badge dot color	string	-	
count	Number to show in badge	ReactNode	-	
dot	Whether to display a red dot instead of `count`	boolean	false	
offset	Set offset of the badge dot	[number, number]	-	
overflowCount	Max count to show	number	99	
showZero	Whether to show badge when `count` is zero	boolean	false	
size	If `count` is set, `size` sets the size of badge	`default` | `small`	-	4.6.0
status	Set Badge as a status dot	`success` | `processing` | `default` | `error` | `warning`	-	
text	If `status` is set, `text` sets the display text of the status `dot`	ReactNode	-	
title	Text to show when hovering over the badge	string	-	

Badge.Ribbon (4.5.0+)

Property	Description	Type	Default	Version
color	Customize Ribbon color	string	-	
placement	The placement of the Ribbon, `start` and `end` follow text direction (RTL or LTR)	`start` | `end`	`end`	
text	Content inside the Ribbon	ReactNode	-	

Design Token

▾ Global Token

Token Name	Description	Type	Default Value
colorBgContainer	Container background color, e.g: default button, input box, etc. Be sure not to	`string`	⬜ #ffffff

Token Name	Description	Type	Default Value
	confuse this with `colorBgElevated`.		
colorBorderBg	Control the color of background border of element.	`string`	`#ffffff`
colorError	Used to represent the visual elements of the operation failure, such as the error Button, error Result component, etc.	`string`	`#ff4d4f`
colorErrorHover	The hover state of the error color.	`string`	`#ff7875`
colorPrimary	Brand color is one of the most direct visual elements to reflect the characteristics and communication of the product. After you have selected the brand color, we will automatically generate a complete color palette and assign it effective design semantics.	`string`	`#1677ff`
colorSuccess	Used to represent the token sequence of operation success, such as Result, Progress and other components will use these map tokens.	`string`	`#52c41a`
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)`
colorTextLightSolid	Control the highlight color of text with background color, such as the text in Primary Button components.	`string`	`#fff`
colorTextPlaceholder	Control the color of placeholder text.	`string`	`rgba(0, 0, 0, 0.25)`
colorWarning	Used to represent the warning map token, such as Notification, Alert, etc. Alert or Control component(like Input) will use these map tokens.	`string`	`#faad14`
borderRadiusSM	SM size border radius, used in small size components, such as Button, Input, Select and other input components in small size	`number`	4
fontFamily	The font family of Ant Design prioritizes the default interface font of the system, and 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.	`string`	-apple-system, BlinkMacSystemFont, 'Segoe UI', 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
fontSizeSM	Small font size	`number`	12
lineHeight	Line height of text.	`number`	1.5714285714285714
lineWidth	Border width of base components	`number`	1

Token Name	Description	Type	Default Value
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
motionDurationSlow	Motion speed, slow speed. Used for large element animation interaction.	`string`	0.3s
motionEaseOutBack	Preset motion curve.	`string`	cubic-bezier(0.12, 0.4, 0.29, 1.46)
paddingXS	Control the extra small padding of the element.	`number`	8