Progress

Display the current progress of an operation flow.

When To Use

If it will take a long time to complete an operation, you can use `Progress` to show the current progress and status.

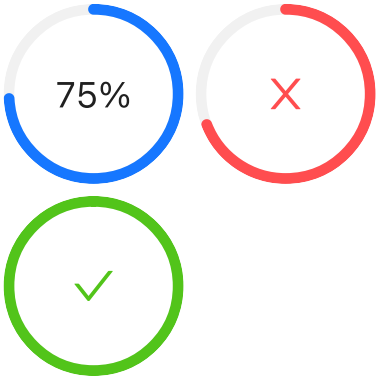
- When an operation will interrupt the current interface, or it needs to run in the background for more than 2 seconds.
- When you need to display the completion percentage of an operation.

Examples

Progress bar

A standard progress bar.

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

const App: React.FC = () => (
  <>
    <Progress percent={30} />
    <Progress percent={50} status="active" />
    <Progress percent={70} status="exception" />
    <Progress percent={100} />
    <Progress percent={50} showInfo={false} />
  </>
);

export default App;
```



Circular progress bar

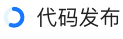
A circular progress bar.

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

const App: React.FC = () => (
  <Space wrap>
    <Progress type="circle" percent={75} />
    <Progress type="circle" percent={70} status="exception" />
    <Progress type="circle" percent={100} />
  </Space>
);

export default App;
```

Mini size progress bar

Appropriate for a narrow area.

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

const App: React.FC = () => (
  <div style={{ width: 170 }}>
    <Progress percent={30} size="small" />
    <Progress percent={50} size="small" status="active" />
    <Progress percent={70} size="small" status="exception" />
    <Progress percent={100} size="small" />
  </div>
);

export default App;
```



Responsive circular progress bar

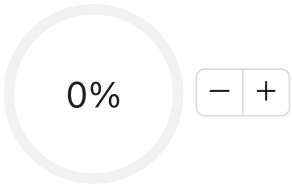
Responsive circular progress bar. When `width` is smaller than 20, progress information will be displayed in Tooltip.

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

const App: React.FC = () => (
  <>
    <Progress
      type="circle"
      trailColor="#e6f4ff"
      percent={60}
      strokeWidth={20}
      size={14}
      format={(number) => `进行中，已完成${number}`}
    />
    <span style={{ marginLeft: 8 }}>代码发布</span>
  </>
);

export default App;
```

Mini size circular progress bar

A smaller circular progress bar.

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

const App: React.FC = () => (
  <Space wrap>
    <Progress type="circle" percent={30} size=
    <Progress type="circle" percent={70} size=
    <Progress type="circle" percent={100} size
  </Space>
);

export default App;
```

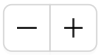


Dynamic circular progress bar

A dynamic progress bar is better.

```
import React, { useState } from 'react';
import { MinusOutlined, PlusOutlined } from '@
import { Button, Progress } from 'antd';

const App: React.FC = () => {
  const [percent, setPercent] = useState<numbe

  const increase = () => {
    setPercent((prevPercent) => {
      const newPercent = prevPercent + 10;
      if (newPercent > 100) {
        return 100;
      }
      return newPercent;
    });
  };

  const decline = () => {
    setPercent((prevPercent) => {
      const newPercent = prevPercent - 10;
      if (newPercent < 0) {
        return 0;
      }
      return newPercent;
    });
  };

  return (
    <>
      <Progress type="circle" percent={percent
      <Button.Group>
        <Button onClick={decline} icon={<Minus
        <Button onClick={increase} icon={<Plus
      </Button.Group>
    </>
  );
};

export default App;
```

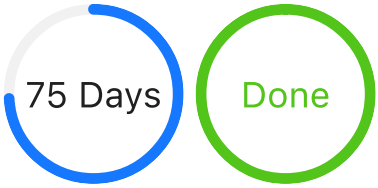
0%



Dynamic

A dynamic progress bar is better.

```
import React, { useState } from 'react';
import { MinusOutlined, PlusOutlined } from '@
import { Button, Progress } from 'antd';

const App: React.FC = () => {
  const [percent, setPercent] = useState<numbe

  const increase = () => {
    setPercent((prevPercent) => {
      const newPercent = prevPercent + 10;
      if (newPercent > 100) {
        return 100;
      }
      return newPercent;
    });
  };

  const decline = () => {
    setPercent((prevPercent) => {
      const newPercent = prevPercent - 10;
      if (newPercent < 0) {
        return 0;
      }
      return newPercent;
    });
  };

  return (
    <>
      <Progress percent={percent} />
      <Button.Group>
        <Button onClick={decline} icon={<Minus
        <Button onClick={increase} icon={<Plus
      </Button.Group>
    </>
  );
};

export default App;
```

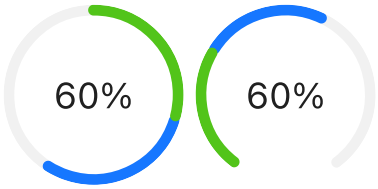


Custom text format

You can set a custom text by setting the `format` prop.

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

const App: React.FC = () => (
  <Space wrap>
    <Progress type="circle" percent={75} formc
    <Progress type="circle" percent={100} form
  </Space>
);

export default App;
```

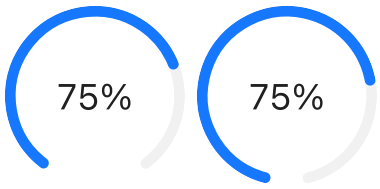
60%



Progress bar with success segment

A standard progress bar. Doesn't support trail color when `type="circle|dashboard"`.

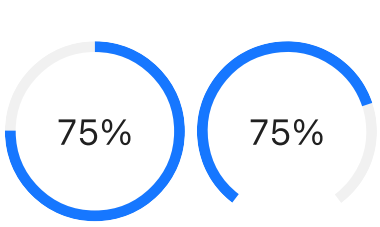
```
import React from 'react';
import { Progress, Tooltip, Space } from 'antc

const App: React.FC = () => (
  <>
    <Tooltip title="3 done / 3 in progress / 4
      <Progress percent={60} success={{ percer
    </Tooltip>
    <Space wrap>
      <Tooltip title="3 done / 3 in progress /
        <Progress percent={60} success={{ perc
      </Tooltip>
      <Tooltip title="3 done / 3 in progress /
        <Progress percent={60} success={{ perc
      </Tooltip>
    </Space>
  </>
);

export default App;
```



Dashboard

By setting `type=dashboard`, you can get a dashboard style of progress easily. Modify `gapDegree` to set the degree of gap.

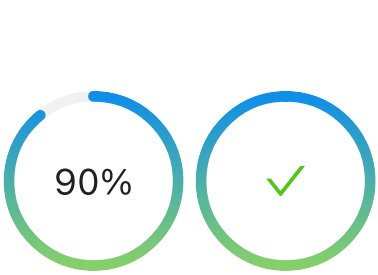
```
import React from 'react';
import { Progress, Space } from 'antd';

const App: React.FC = () => (
  <Space wrap>
    <Progress type="dashboard" percent={75} />
    <Progress type="dashboard" percent={75} ga
  </Space>
);

export default App;
```



Stroke Linecap

By setting `strokeLinecap="butt"`, you can change the linecaps from `round` to `butt`, see stroke-linecap for more information.

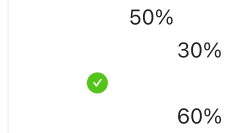
```
import React from 'react';
import { Progress, Space } from 'antd';

const App: React.FC = () => (
  <>
    <Progress strokeLinecap="butt" percent={75
    <Space wrap>
      <Progress strokeLinecap="butt" type="cir
      <Progress strokeLinecap="butt" type="das
    </Space>
  </>
);

export default App;
```



Custom line gradient

A package of `linear-gradient`. It is recommended to only pass two colors.

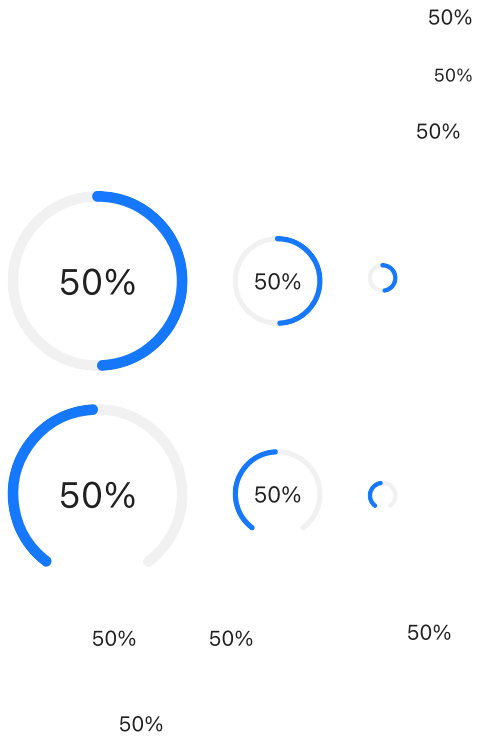
```
import React from 'react';
import { Progress, Space } from 'antd';

const App: React.FC = () => (
  <>
    <Progress percent={99.9} strokeColor={{ '0
    <Progress percent={99.9} status="active" s
    <Space wrap>
      <Progress type="circle" percent={90} str
      <Progress type="circle" percent={100} st
    </Space>
  </>
);

export default App;
```



Progress bar with steps

A progress bar with steps.

```
import React from 'react';
import { Progress } from 'antd';
import { red, green } from '@ant-design/colors

const App: React.FC = () => (
  <>
    <Progress percent={50} steps={3} />
    <br />
    <Progress percent={30} steps={5} />
    <br />
    <Progress percent={100} steps={5} size="sm
    <br />
    <Progress percent={60} steps={5} strokeCol
  </>
);

export default App;
```



Progress size

The size of progress.

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

const App: React.FC = () => (
  <>
    <Space direction="vertical">
      <Progress percent={50} />
      <Progress percent={50} size="small" />
      <Progress percent={50} size={[300, 20]}
    </Space>
    <br />
    <br />
    <Space size={30}>
      <Progress type="circle" percent={50} />
      <Progress type="circle" percent={50} siz
      <Progress type="circle" percent={50} siz
    </Space>
    <br />
    <br />
    <Space size={30}>
      <Progress type="dashboard" percent={50}
      <Progress type="dashboard" percent={50}
      <Progress type="dashboard" percent={50}
    </Space>
    <br />
    <br />
    <Space size={30} wrap>
      <Progress steps={3} percent={50} />
      <Progress steps={3} percent={50} size="s
      <Progress steps={3} percent={50} size={2
      <Progress steps={3} percent={50} size={[
    </Space>
  </>
);

export default App;
```

API

Properties that shared by all types.

Property	Description	Type	Default	Version
format	The template function of the content	function(percent, successPercent)	(percent) => percent + `%`	-
percent	To set the completion percentage	number	0	-
showInfo	Whether to display the progress value and the status icon	boolean	true	
status	To set the status of the Progress, options: `success` `exception` `normal` `active` (line only)	string	-	
strokeColor	The color of progress bar	string	-	-
strokeLinecap	To set the style of the progress linecap	`round` | `butt` | `square`, see stroke-linecap	`round`	-
success	Configs of successfully progress bar	{ percent: number, strokeColor: string }	-	-
trailColor	The color of unfilled part	string	-	-
type	To set the type, options: `line` `circle` `dashboard`	string	`line`	
size	Progress size	number | [number, number] | "small" | "default"	"default"	v5.3.0

`type="line"`

Property	Description	Type	Default	Version
steps	The total step count	number	-	-
strokeColor	The color of progress bar, render `linear-gradient` when passing an object, could accept `string[]` when has `steps`.	string | string[] | { from: string; to: string; direction: string }	-	4.21.0: `string[]`

`type="circle"`

Property	Description	Type	Default	Version
strokeColor	The color of circular progress, render `linear-gradient` when passing an object	string | object	-	-
strokeWidth	To set the width of the circular progress, unit: percentage of the canvas width	number	6	-

`type="dashboard"`

Property	Description	Type	Default
gapDegree	The gap degree of half circle, 0 ~ 295	number	75
gapPosition	The gap position, options: `top` `bottom` `left` `right`	string	`bottom`
strokeWidth	To set the width of the dashboard progress, unit: percentage of the canvas width	number	6

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 confuse this with `colorBgElevated`.	`string`	`#ffffff`
colorError	Used to represent the visual elements of the operation failure, such as the error Button, error Result component, etc.	`string`	`#ff4d4f`
colorFillSecondary	The second level of fill color can outline the shape of the element more clearly, such as Rate, Skeleton, etc. It can also be used as the Hover state of the third level of fill color, such as Table, etc.	`string`	`rgba(0, 0, 0, 0.06)`
colorInfo	Used to represent the operation information of the Token sequence, such as Alert, Tag, Progress, and other components use these map tokens.	`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)`

Token Name	Description	Type	Default Value
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
marginXS	Control the margin of an element, with a small size.	`number`	8
marginXXS	Control the margin of an element, with the smallest size.	`number`	4
motionDurationSlow	Motion speed, slow speed. Used for large element animation interaction.	`string`	0.3s
motionEaseInOutCirc	Preset motion curve.	`string`	cubic-bezier(0.78, 0.14, 0.15, 0.86)
motionEaseOutQuint	Preset motion curve.	`string`	cubic-bezier(0.23, 1, 0.32, 1)
paddingXS	Control the extra small padding of the element.	`number`	8