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

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

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

export default App;
```

Circular progress bar

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

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

export default App;
```

Mini size progress bar

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

const App: React.FC = () => (
  <Flex vertical gap="small" style={{ width: 180 }}>
    <Progress percent={30} size="small" />
    <Progress percent={50} size="small" status="active" />
    <Progress percent={70} size="small" status="exception" />
    <Progress percent={100} size="small" />
  </Flex>
);

export default App;
```

Responsive circular progress bar

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

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

export default App;
```

Mini size circular progress bar

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

const App: React.FC = () => (
  <Flex wrap gap="small">
    <Progress type="circle" percent={30} size={80} />
    <Progress type="circle" percent={70} size={80} status="exception" />
    <Progress type="circle" percent={100} size={80} />
  </Flex>
);

export default App;
```

Dynamic

```
import React, { useState } from 'react';
import { MinusOutlined, PlusOutlined } from '@ant-design/icons';
import { Button, Flex, Progress, Space } from 'antd';

const App: React.FC = () => {
  const [percent, setPercent] = useState<number>(0);

  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 (
    <Flex vertical gap="small">
      <Flex vertical gap="small">
        <Progress percent={percent} type="line" />
        <Progress percent={percent} type="circle" />
      </Flex>
      <Space.Compact>
        <Button onClick={decline} icon={<MinusOutlined />} />
        <Button onClick={increase} icon={<PlusOutlined />} />
      </Space.Compact>
    </Flex>
  );
};

export default App;
```

Custom text format

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

const App: React.FC = () => (
  <Flex gap="small" wrap>
    <Progress type="circle" percent={75} format={(percent) => `${percent}
Days`} />
    <Progress type="circle" percent={100} format={() => 'Done'} />
  </Flex>
);

export default App;
```

Dashboard

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

const App: React.FC = () => (
  <Flex gap="small" wrap>
    <Progress type="dashboard" percent={75} />
    <Progress type="dashboard" percent={75} gapDegree={30} />
  </Flex>
);

export default App;
```

Progress bar with success segment

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

const App: React.FC = () => (
  <Flex gap="small" vertical>
    <Tooltip title="3 done / 3 in progress / 4 to do">
      <Progress percent={60} success={{ percent: 30 }} />
    </Tooltip>
    <Flex gap="small" wrap>
      <Tooltip title="3 done / 3 in progress / 4 to do">
        <Progress percent={60} success={{ percent: 30 }} type="circle" />
      </Tooltip>
      <Tooltip title="3 done / 3 in progress / 4 to do">
        <Progress percent={60} success={{ percent: 30 }} type="dashboard"
/>
      </Tooltip>
    </Flex>
  </Flex>
);

export default App;
```

Stroke Linecap

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

const App: React.FC = () => (
  <Flex vertical gap="small">
    <Progress strokeLinecap="butt" percent={75} />
    <Flex wrap gap="small">
      <Progress strokeLinecap="butt" type="circle" percent={75} />
      <Progress strokeLinecap="butt" type="dashboard" percent={75} />
    </Flex>
  </Flex>
);

export default App;
```

Custom line gradient

```
import React from 'react';
import { Flex, Progress } from 'antd';
import type { ProgressProps } from 'antd';

const twoColors: ProgressProps['strokeColor'] = {
  '0%': '#108ee9',
  '100%': '#87d068',
};

const conicColors: ProgressProps['strokeColor'] = {
  '0%': '#87d068',
  '50%': '#ffe58f',
  '100%': '#ffccc7',
};

const App: React.FC = () => (
  <Flex vertical gap="middle">
    <Progress percent={99.9} strokeColor={twoColors} />
    <Progress percent={50} status="active" strokeColor={{ from: '#108ee9',
to: '#87d068' }} />
    <Flex gap="small" wrap>
      <Progress type="circle" percent={90} strokeColor={twoColors} />
      <Progress type="circle" percent={100} strokeColor={twoColors} />
      <Progress type="circle" percent={93} strokeColor={conicColors} />
    </Flex>
    <Flex gap="small" wrap>
      <Progress type="dashboard" percent={90} strokeColor={twoColors} />
      <Progress type="dashboard" percent={100} strokeColor={twoColors} />
      <Progress type="dashboard" percent={93} strokeColor={conicColors} />
    </Flex>
  </Flex>
);

export default App;
```

Progress bar with steps

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

const App: React.FC = () => (
  <Flex gap="small" vertical>
    <Progress percent={50} steps={3} />
    <Progress percent={30} steps={5} />
    <Progress percent={100} steps={5} size="small" strokeColor={green[6]}
/>
    <Progress percent={60} steps={5} strokeColor={[green[6], green[6],
red[5]]} />
  </Flex>
);

export default App;
```

Circular progress bar with steps

v5.16.0

```
import React from 'react';
import { Flex, Progress, Slider, Typography } from 'antd';

const App: React.FC = () => {
  const [stepsCount, setStepsCount] = React.useState<number>(5);
  const [stepsGap, setStepsGap] = React.useState<number>(7);
  return (
    <>
      <Typography.Title level={5}>Custom count:</Typography.Title>
      <Slider min={2} max={10} value={stepsCount} onChange={setStepsCount}
/>
      <Typography.Title level={5}>Custom gap:</Typography.Title>
      <Slider step={4} min={0} max={40} value={stepsGap} onChange=
{setStepsGap} />
      <Flex wrap gap="middle" style={{ marginTop: 16 }}>
        <Progress
          type="dashboard"
          steps={8}
          percent={50}
          trailColor="rgba(0, 0, 0, 0.06)"
          strokeWidth={20}
        />
        <Progress
          type="circle"
          percent={100}
          steps={{ count: stepsCount, gap: stepsGap }}
          trailColor="rgba(0, 0, 0, 0.06)"
          strokeWidth={20}
        />
      </Flex>
    </>
  );
};

export default App;
```

Progress size

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

const App: React.FC = () => (
  <Flex vertical gap="middle">
    <Flex vertical gap="small" style={{ width: 300 }}>
      <Progress percent={50} />
      <Progress percent={50} size="small" />
      <Progress percent={50} size={[300, 20]} />
    </Flex>
    <Flex align="center" wrap gap={30}>
      <Progress type="circle" percent={50} />
      <Progress type="circle" percent={50} size="small" />
      <Progress type="circle" percent={50} size={20} />
    </Flex>
    <Flex align="center" wrap gap={30}>
      <Progress type="dashboard" percent={50} />
      <Progress type="dashboard" percent={50} size="small" />
      <Progress type="dashboard" percent={50} size={20} />
    </Flex>
    <Flex align="center" wrap gap={30}>
      <Progress steps={3} percent={50} />
      <Progress steps={3} percent={50} size="small" />
      <Progress steps={3} percent={50} size={20} />
      <Progress steps={3} percent={50} size={[20, 30]} />
    </Flex>
  </Flex>
);

export default App;
```

Change progress value position

v5.18.0

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

const App: React.FC = () => (
  <Flex gap="small" vertical>
    <Progress
      percent={0}
      percentPosition={{ align: 'center', type: 'inner' }}
      size={[200, 20]}
      strokeColor="#E6F4FF"
    />
    <Progress percent={10} percentPosition={{ align: 'center', type:
'inner' }} size={[300, 20]} />
    <Progress
      percent={50}
      percentPosition={{ align: 'start', type: 'inner' }}
      size={[300, 20]}
      strokeColor="#B7EB8F"
    />
    <Progress
      percent={60}
      percentPosition={{ align: 'end', type: 'inner' }}
      size={[300, 20]}
      strokeColor="#001342"
    />
    <Progress percent={100} percentPosition={{ align: 'center', type:
'inner' }} size={[400, 20]} />
    <Progress percent={60} percentPosition={{ align: 'start', type: 'outer'
}} />
    <Progress percent={100} percentPosition={{ align: 'start', type:
'outer' }} />
    <Progress percent={60} percentPosition={{ align: 'center', type:
'outer' }} size="small" />
    <Progress percent={100} percentPosition={{ align: 'center', type:
'outer' }} />
  </Flex>
);

export default App;
```

API

Common props ref: Common props

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 | string, number] | { width: number, height: number } | "small" | "default"	"default"	5.3.0, Object: 5.18.0

`type="line"`

Property	Description	Type	Default	Version
steps	The total step count	number	-	-
rounding	The function to round the value	(step: number) => number	Math.round	5.24.0
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[]`
percentPosition	Progress value position, passed in object, `align` indicates the horizontal position of the value, `type` indicates whether the value is inside or outside the progress bar	{ align: string; type: string }	{ align: "end", type: "outer" }	5.18.0

`type="circle"`

Property	Description	Type	Default	Version
steps	The total step count.When passing an object, `count` refers to the number of steps, and `gap` refers to the distance between them.When passing number, the default value for `gap` is 2.	number | { count: number, gap: number }	-	5.16.0
strokeColor	The color of circular progress, render gradient when passing an object	string | { number%: string }	-	-
strokeWidth	To set the width of the circular progress, unit: percentage of the canvas width	number	6	-

`type="dashboard"`

Property	Description	Type	Default	Version
steps	The total step count.When passing an object, `count` refers to the number of steps, and `gap` refers to the distance between them.When passing number, the default value for `gap` is 2.	number | { count: number, gap: number }	-	5.16.0
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