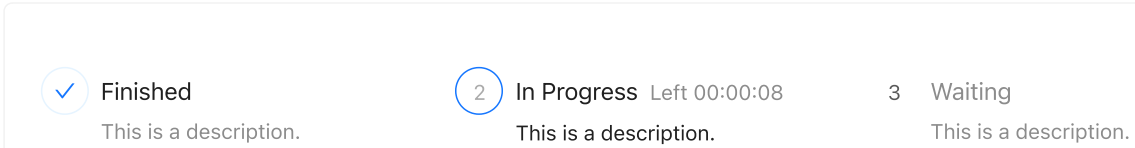
Steps

`Steps` is a navigation bar that guides users through the steps of a task.

When To Use

When a given task is complicated or has a certain sequence in the series of subtasks, we can decompose it into several steps to make things easier.

Examples



Basic

The most basic step bar.

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

const description = 'This is a description.';
const App: React.FC = () => (
  <Steps
    current={1}
    items={[
      {
        title: 'Finished',
        description,
      },
      {
        title: 'In Progress',
        description,
        subTitle: 'Left 00:00:08',
      },
      {
        title: 'Waiting',
        description,
      },
    ]}
  />
);

export default App;
```

Mini version

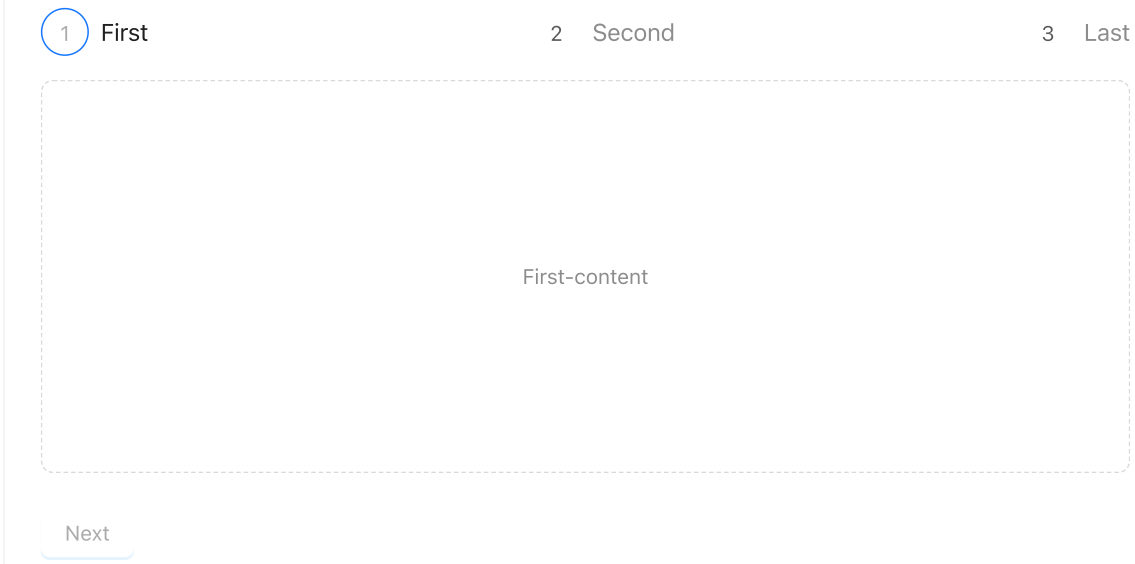
By setting like this: `<Steps size="small">`, you can get a mini version.

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

const App: React.FC = () => (
  <Steps
    size="small"
    current={1}
    items={[
      {
        title: 'Finished',
      },
      {
        title: 'In Progress',
      },
      {
        title: 'Waiting',
      },
    ]}
  />
);

export default App;
```

With icon

You can use your own custom icons by setting the property `icon` for `items`.

```
import React from 'react';
import { LoadingOutlined, SmileOutlined, SolutionOutlined, UserOutlined } from '@ant-design/icon
import { Steps } from 'antd';

const App: React.FC = () => (
  <Steps
    items={[
      {
        title: 'Login',
        status: 'finish',
        icon: <UserOutlined />,
      },
      {
        title: 'Verification',
        status: 'finish',
        icon: <SolutionOutlined />,
      },
      {
        title: 'Pay',
        status: 'process',
        icon: <LoadingOutlined />,
      },
      {
        title: 'Done',
        status: 'wait',
        icon: <SmileOutlined />,
      },
    ]}
  />
);

export default App;
```



Switch Step

Cooperate with the content and buttons, to represent the progress of a process.

```
import React, { useState } from 'react';
import { Button, message, Steps, theme } from 'antd';

const steps = [
  {
    title: 'First',
    content: 'First-content',
  },
  {
    title: 'Second',
    content: 'Second-content',
  },
  {
    title: 'Last',
    content: 'Last-content',
  },
];

const App: React.FC = () => {
  const { token } = theme.useToken();
  const [current, setCurrent] = useState(0);

  const next = () => {
    setCurrent(current + 1);
  };

  const prev = () => {
    setCurrent(current - 1);
  };

  const items = steps.map((item) => ({ key: item.title, title: item.title }));

  const contentStyle: React.CSSProperties = {
    lineHeight: '260px',
    textAlign: 'center',
    color: token.colorTextTertiary,
    backgroundColor: token.colorFillAlter,
    borderRadius: token.borderRadiusLG,
    border: `1px dashed ${token.colorBorder}`,
    marginTop: 16,
  };

  return (
    <>
      <Steps current={current} items={items} />
      <div style={contentStyle}>{steps[current].content}</div>
      <div style={{ marginTop: 24 }}>
```

Vertical

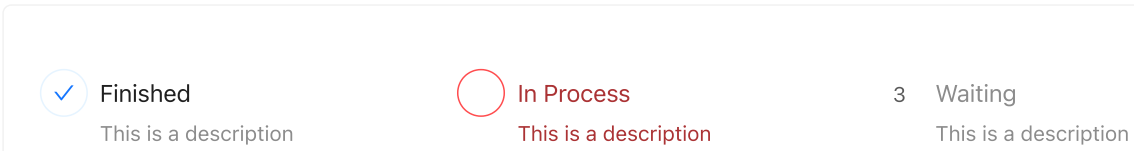
A simple step bar in the vertical direction.

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

const description = 'This is a description.';
const App: React.FC = () => (
  <Steps
    direction="vertical"
    current={1}
    items={[
      {
        title: 'Finished',
        description,
      },
      {
        title: 'In Progress',
        description,
      },
      {
        title: 'Waiting',
        description,
      },
    ]}
  />
);

export default App;
```

Vertical mini version

A simple mini version step bar in the vertical direction.

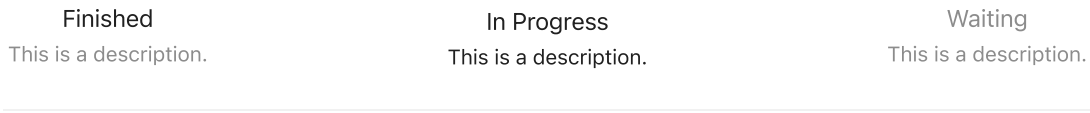
```
import React from 'react';
import { Steps } from 'antd';

const description = 'This is a description.';
const App: React.FC = () => (
  <Steps
    direction="vertical"
    size="small"
    current={1}
    items={[
      { title: 'Finished', description },
      {
        title: 'In Progress',
        description,
      },
      {
        title: 'Waiting',
        description,
      },
    ]}
  />
);

export default App;
```

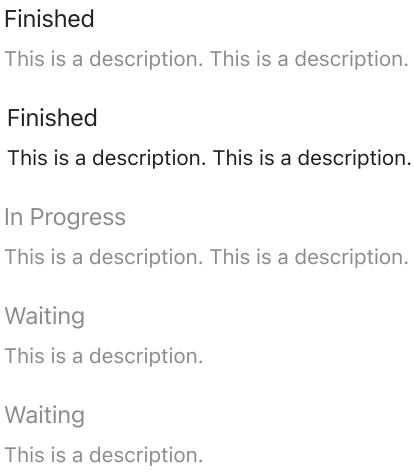


Error status

By using `status` of `Steps`, you can specify the state for current step.

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

const description = 'This is a description';
const App: React.FC = () => (
  <Steps
    current={1}
    status="error"
    items={[
      {
        title: 'Finished',
        description,
      },
      {
        title: 'In Process',
        description,
      },
      {
        title: 'Waiting',
        description,
      },
    ]}
  />
);

export default App;
```

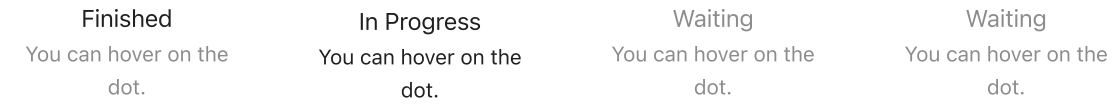




Dot Style

Steps with progress dot style.

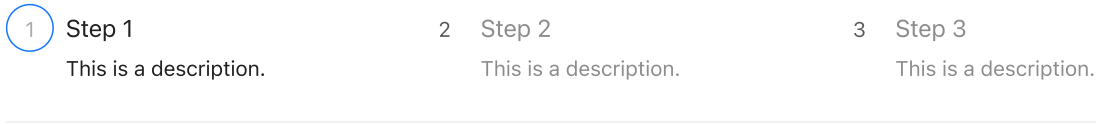
```
import React from 'react';
import { Divider, Steps } from 'antd';

const App: React.FC = () => (
  <>
    <Steps
      progressDot
      current={1}
      items={[
        {
          title: 'Finished',
          description: 'This is a description.',
        },
        {
          title: 'In Progress',
          description: 'This is a description.',
        },
        {
          title: 'Waiting',
          description: 'This is a description.',
        },
      ]}
    />
    <Divider />
    <Steps
      progressDot
      current={1}
      direction="vertical"
      items={[
        {
          title: 'Finished',
          description: 'This is a description. This is a description.',
        },
        {
          title: 'Finished',
          description: 'This is a description. This is a description.',
        },
        {
          title: 'In Progress',
          description: 'This is a description. This is a description.',
        },
        {
          title: 'Waiting',
```

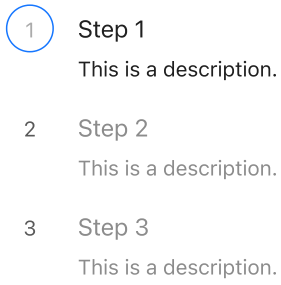


Customized Dot Style

You can customize the display for Steps with progress dot style.

```
import React from 'react';
import type { StepsProps } from 'antd';
import { Popover, Steps } from 'antd';

const customDot: StepsProps['progressDot'] = (dot, { status, index }) => (
  <Popover
    content={
      <span>
        step {index} status: {status}
      </span>
    }
  >
    {dot}
  </Popover>
);
const description = 'You can hover on the dot.';
const App: React.FC = () => (
  <Steps
    current={1}
    progressDot={customDot}
    items={[
      {
        title: 'Finished',
        description,
      },
      {
        title: 'In Progress',
        description,
      },
      {
        title: 'Waiting',
        description,
      },
      {
        title: 'Waiting',
        description,
      },
    ]}
  />
);

export default App;
```

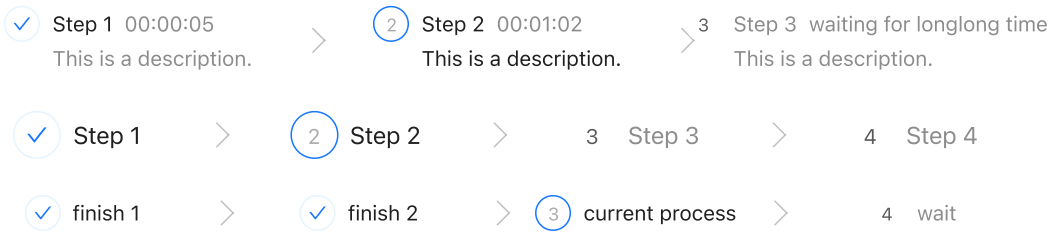




Clickable

Setting `onChange` makes Steps clickable.

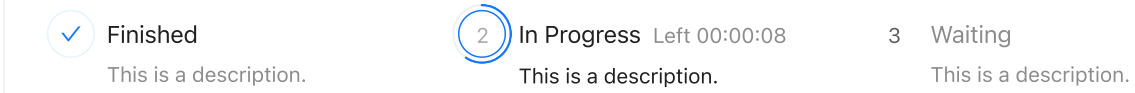
```
import React, { useState } from 'react';
import { Divider, Steps } from 'antd';

const App: React.FC = () => {
  const [current, setCurrent] = useState(0);

  const onChange = (value: number) => {
    console.log('onChange:', value);
    setCurrent(value);
  };
  const description = 'This is a description.';

  return (
    <>
      <Steps
        current={current}
        onChange={onChange}
        items={[
          {
            title: 'Step 1',
            description,
          },
          {
            title: 'Step 2',
            description,
          },
          {
            title: 'Step 3',
            description,
          },
        ]}
      />

      <Divider />

      <Steps
        current={current}
        onChange={onChange}
        direction="vertical"
        items={[
          {
            title: 'Step 1',
            description,
          },
          {
            title: 'Step 2',
            description,
          },
          {
            title: 'Step 3',
```



Navigation Steps

Navigation steps.

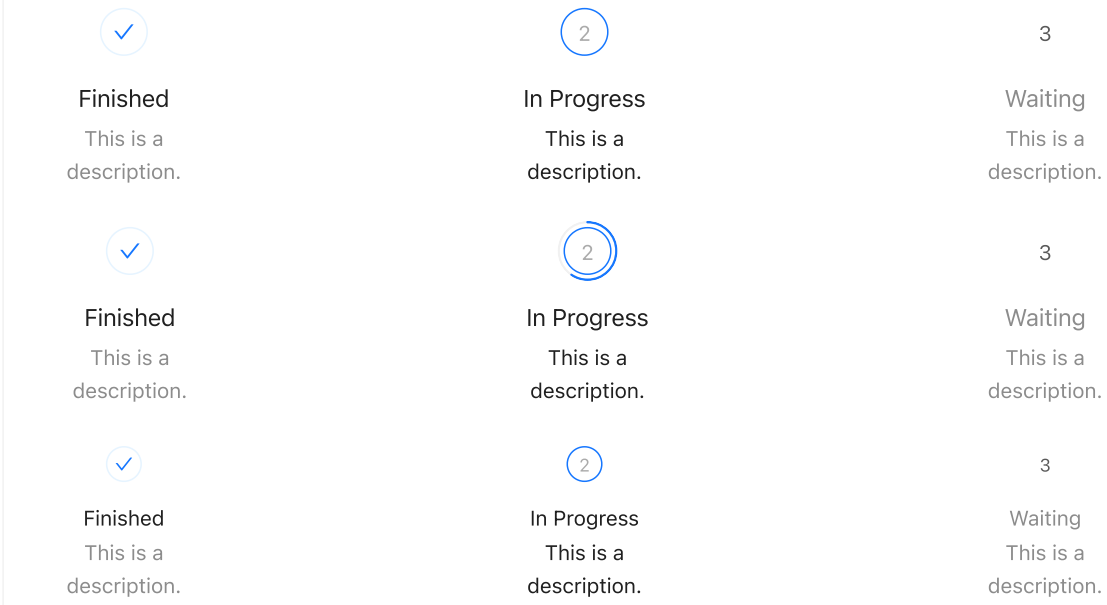
```
import React, { useState } from 'react';
import { Steps } from 'antd';

const App: React.FC = () => {
  const [current, setCurrent] = useState(0);

  const onChange = (value: number) => {
    console.log('onChange:', value);
    setCurrent(value);
  };

  return (
    <>
      <Steps
        type="navigation"
        size="small"
        current={current}
        onChange={onChange}
        className="site-navigation-steps"
        items={[
          {
            title: 'Step 1',
            subTitle: '00:00:05',
            status: 'finish',
            description: 'This is a description.',
          },
          {
            title: 'Step 2',
            subTitle: '00:01:02',
            status: 'process',
            description: 'This is a description.',
          },
          {
            title: 'Step 3',
            subTitle: 'waiting for longlong time',
            status: 'wait',
            description: 'This is a description.',
          },
        ]}
      />
      <Steps
        type="navigation"
        current={current}
        onChange={onChange}
        className="site-navigation-steps"
        items={[
          {
            status: 'finish',
            title: 'Step 1',
          },
          {
            status: 'process',
            title: 'Step 2',
          },
          {
            status: 'wait',
            title: 'Step 3',
```



Steps with progress

Steps with progress.

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

const description = 'This is a description.';
const App: React.FC = () => (
  <Steps
    current={1}
    percent={60}
    items={[
      {
        title: 'Finished',
        description,
      },
      {
        title: 'In Progress',
        subTitle: 'Left 00:00:08',
        description,
      },
      {
        title: 'Waiting',
        description,
      },
    ]}
  />
);

export default App;
```



Label Placement

Set labelPlacement to `vertical`.

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

const description = 'This is a description.';
const items = [
  {
    title: 'Finished',
    description,
  },
  {
    title: 'In Progress',
    description,
  },
  {
    title: 'Waiting',
    description,
  },
];
const App: React.FC = () => (
  <>
    <Steps current={1} labelPlacement="vertical" items={items} />
    <br />
    <Steps current={1} percent={60} labelPlacement="vertical" items={items} />
    <br />
    <Steps current={1} size="small" labelPlacement="vertical" items={items} />
  </>
);

export default App;
```

Ant Design Title 1

Ant Design, a design language for background applications, is refined by Ant UED Team

Step 1 Step 2 Step 3

Ant Design Title 2

Ant Design, a design language for background applications, is refined by Ant UED Team

Step 1 Step 2 Step 3

Ant Design Title 3

Ant Design, a design language for background applications, is refined by Ant UED Team

Step 1 Step 2 Step 3

Ant Design Title 4

Ant Design, a design language for background applications, is refined by Ant UED Team

Step 1 Step 2 Step 3

Inline Steps

Inline type steps, suitable for displaying the process and current state of the object in the list content scene.

```
import type { StepsProps } from 'antd';
import { Avatar, List, Steps } from 'antd';
import React from 'react';

const data = [
  {
    title: 'Ant Design Title 1',
    current: 0,
  },
  {
    title: 'Ant Design Title 2',
    current: 1,
    status: 'error',
  },
  {
    title: 'Ant Design Title 3',
    current: 2,
  },
  {
    title: 'Ant Design Title 4',
    current: 1,
  },
];

const items = [
  {
    title: 'Step 1',
    description: 'This is a Step 1.',
  },
  {
    title: 'Step 2',
    description: 'This is a Step 2.',
  },
  {
    title: 'Step 3',
    description: 'This is a Step 3.',
  },
];

const App: React.FC = () => (
  <div>
    <List
      itemLayout="horizontal"
      dataSource={data}
      renderItem={(item, index) => (
        <List.Item>
```

```
          <List.Item.Meta
            avatar={
              <Avatar src={`https://xsgames.co/randomusers/avatar.php?g=pixel&key=${index}`} />
            }
            title={<a href="https://ant.design">{item.title}</a>}
            description="Ant Design, a design language for background applications, is refined b
          />
          <Steps
            style={{ marginTop: 8 }}
            type="inline"
            current={item.current}
```

Property	Description	Type	Default	Version
className	Additional class to Steps	string	-	
current	To set the current step, counting from 0. You can overwrite this state by using `status` of `Step`	number	0	
direction	To specify the direction of the step bar, `horizontal` or `vertical`	string	`horizontal`	
initial	Set the initial step, counting from 0	number	0	
labelPlacement	Place title and description with `horizontal` or `vertical` direction	string	`horizontal`	
percent	Progress circle percentage of current step in `process` status (only works on basic Steps)	number	-	4.5.0
progressDot	Steps with progress dot style, customize the progress dot by setting it to a function. labelPlacement will be `vertical`	boolean | (iconDot, {index, status, title, description}) => ReactNode	false	
responsive	Change to vertical direction when screen width smaller than `532px`	boolean	true	
size	To specify the size of the step bar, `default` and `small` are currently supported	string	`default`	
status	To specify the status of current step, can be set to one of the following values: `wait` `process` `finish` `error`	string	`process`	

Property	Description	Type	Default	Version
type	Type of steps, can be set to one of the following values: `default` `navigation` `inline`	string	`default`	inline: 5.0
onChange	Trigger when Step is changed	(current) => void	-	
items	StepItem content	StepItem	[]	4.24.0

`type="inline"`

Property	Description	Type	Default	Version
className	Additional class to Steps	string	-	
current	To set the current step, counting from 0. You can overwrite this state by using `status` of `Step`	number	0	
initial	Set the initial step, counting from 0	number	0	
status	To specify the status of current step, can be set to one of the following values: `wait` `process` `finish` `error`	string	`process`	
onChange	Trigger when Step is changed	(current) => void	-	
items	StepItem content. not supported: `icon` `subtitle`	StepItem	[]	4.24.0

StepItem

A single step in the step bar.

Property	Description	Type	Default	Version
description	Description of the step, optional property	ReactNode	-	
disabled	Disable click	boolean	false	
icon	Icon of the step, optional property	ReactNode	-	
status	To specify the status. It will be automatically set by `current` of `Steps` if not configured. Optional values are: `wait` `process` `finish` `error`	string	`wait`	
subTitle	Subtitle of the step	ReactNode	-	

Property	Description	Type	Default	Version
title	Title of the step	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 confuse this with `colorBgElevated`.	string	#ffffff
colorBorderBg	Control the color of background border of element.	string	#ffffff
colorBorderSecondary	Slightly lighter than the default border color, this color is the same as `colorSplit`. Solid color is used.	string	#f0f0f0
colorError	Used to represent the visual elements of the operation failure, such as the error Button, error Result component, etc.	string	#ff4d4f
colorFillContent	Control the background color of content area.	string	rgba(0, 0, 0, 0.06)
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
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)
colorTextDescription	Control the font color of text description.	string	rgba(0, 0, 0, 0.45)
colorTextDisabled	Control the color of text in disabled state.	string	rgba(0, 0, 0, 0.25)
colorTextLabel	Control the font color of text label.	string	rgba(0, 0, 0, 0.65)
colorTextLightSolid	Control the highlight color of text with background color, such as the text in Primary Button components.	string	#fff
colorTextQuaternary	The fourth level of text color is the lightest text color, such as form input prompt text, disabled color text, etc.	string	rgba(0, 0, 0, 0.25)
borderRadiusSM	SM size border radius, used in small size components, such as Button, Input, Select and other input components in small size	number	4

Token Name	Description	Type	Default Value
controlHeight	The height of the basic controls such as buttons and input boxes in Ant Design	`number`	32
controlHeightLG	LG component height	`number`	40
controlHeightSM	SM component height	`number`	24
controlItemBgActive	Control the background color of control component item when active.	`string`	`#e6f4ff`
controlItemBgHover	Control the background color of control component item when hovering.	`string`	`rgba(0, 0, 0, 0.04)`
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
fontSizeHeading3	Font size of h3 tag.	`number`	24
fontSizeIcon	Control the font size of operation icon in Select, Cascader, etc. Normally same as fontSizeSM.	`number`	12
fontSizeLG	Large font size	`number`	16
fontSizeSM	Small font size	`number`	12
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
lineHeightSM	Line height of small text.	`number`	1.6666666666666667
lineType	Border style of base components	`string`	solid
lineWidth	Border width of base components	`number`	1
lineWidthBold	The default line width of the outline class components, such as Button, Input, Select, etc.	`number`	2
margin	Control the margin of an element, with a medium size.	`number`	16
marginLG	Control the margin of an element, with a large size.	`number`	24

Token Name	Description	Type	Default Value
marginSM	Control the margin of an element, with a medium-small size.	`number`	12
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
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
padding	Control the padding of the element.	`number`	16
paddingLG	Control the large padding of the element.	`number`	24
paddingSM	Control the small padding of the element.	`number`	12
paddingXS	Control the extra small padding of the element.	`number`	8
paddingXXS	Control the extra extra small padding of the element.	`number`	4
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