

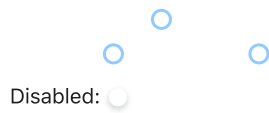
Slider

A Slider component for displaying current value and intervals in range.

When To Use

To input a value in a range.

Examples



Basic

Basic slider. When `range` is `true`, display as dual thumb mode. When `disable` is `true`, the slider will not be interactable.

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

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

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

  return (
    <>
      <Slider defaultValue={30} disabled={disabled}>
        <Slider range defaultValue={[20, 50]} disabled={disabled} />
      </Slider>
      Disabled: <Switch size="small" checked={disabled} />
    </>
  );
};

export default App;
```



Slider with icon

You can add an icon beside the slider to make it meaningful.

```
import React, { useState } from 'react';
import { FrownOutlined, SmileOutlined } from 'antd';
import { Slider } from 'antd';

interface IconSliderProps {
  max: number;
  min: number;
}

const IconSlider: React.FC<IconSliderProps> =
  ({ max, min }) => {
    const [value, setValue] = useState(0);

    const mid = Number(((max - min) / 2).toFixed(1));
    const preColorCls = value >= mid ? 'icon-wr' : 'icon-wr';
    const nextColorCls = value >= mid ? 'icon-wr' : 'icon-wr';

    return (
      <div className="icon-wrapper">
        <FrownOutlined className={preColorCls} />
        <Slider {...props} onChange={setValue} />
        <SmileOutlined className={nextColorCls} />
      </div>
    );
  };

const App: React.FC = () => <IconSlider min={0} max={100} />;

export default App;
```



Slider with InputNumber

Synchronize with [InputNumber](#) component.

```
import React, { useState } from 'react';
import { Col, InputNumber, Row, Slider, Space } from 'antd';

const IntegerStep: React.FC = () => {
  const [inputValue, setInputValue] = useState(1);

  const onChange = (newValue: number) => {
    setInputValue(newValue);
  };

  return (
    <Row>
      <Col span={12}>
        <Slider
          min={1}
          max={20}
          onChange={onChange}
          value={typeof inputValue === 'number' ? inputValue : 1}
        />
      </Col>
      <Col span={4}>
        <InputNumber
          min={1}
          max={20}
          style={{ margin: '0 16px' }}
          value={inputValue}
          onChange={onChange}
        />
      </Col>
    </Row>
  );
};

const DecimalStep: React.FC = () => {
  const [inputValue, setInputValue] = useState(0.01);

  const onChange = (value: number) => {
    if (isNaN(value)) {
      return;
    }
    setInputValue(value);
  };

  return (
    <Row>
      <Col span={12}>
        <Slider
          min={0}
          max={1}
          onChange={onChange}
          value={typeof inputValue === 'number' ? inputValue : 0}
          step={0.01}
        />
      </Col>
      <Col span={4}>
        <InputNumber
          min={0}
          max={1}
          style={{ margin: '0 16px' }}
          step={0.01}
          value={inputValue}
          onChange={onChange}
        />
      </Col>
    </Row>
  );
};
```



Event [✎](#)

The `onChange` callback function will fire when the user changes the slider's value. The

`onAfterChange` callback function will fire when `onmouseup` fired.

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

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

const onAfterChange = (value: number | [number, number]) => {
  console.log('onAfterChange: ', value);
};

const App: React.FC = () => (
  <>
    <Slider defaultValue={30} onChange={onChange} onAfterChange={onAfterChange} />
  </>
);

export default App;
```



Customize tooltip [✎](#)

Use `tooltip.formatter` to format content of `Tooltip`. If `tooltip.formatter` is null, hide it.

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

const formatter = (value: number) => `${value}`;

const App: React.FC = () => (
  <>
    <Slider tooltip={{ formatter }} />
    <Slider tooltip={{ formatter: null }} />
  </>
);

export default App;
```

100°C



Vertical

The vertical Slider.

```
import React from 'react';
import { Slider } from 'antd';
import type { SliderMarks } from 'antd/es/slider';

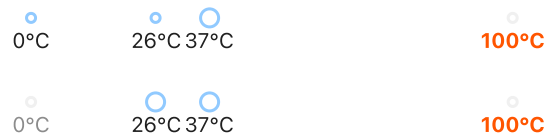
const style: React.CSSProperties = {
  display: 'inline-block',
  height: 300,
  marginLeft: 70,
};

const marks: SliderMarks = {
  0: '0°C',
  26: '26°C',
  37: '37°C',
  100: {
    style: { color: '#f50' },
    label: <strong>100°C</strong>,
  },
};

const App: React.FC = () => (
  <div style={style}>
    <Slider vertical defaultValue={30} />
  </div>
  <div style={style}>
    <Slider vertical range step={10} default
  </div>
  <div style={style}>
    <Slider vertical range marks={marks} def
  </div>
</>
);

export default App;
```

included=true



included=false



marks & step



step=null



Graduated slider


Using `marks` property to mark a graduated slider, use `value` or `defaultValue` to specify the position of thumb. When `included` is false, means that different thumbs are coordinative. when `step` is null, users can only slide the thumbs onto marks.

```
import React from 'react';
import { Slider } from 'antd';
import type { SliderMarks } from 'antd/es/slider';

const marks: SliderMarks = {
  0: '0°C',
  26: '26°C',
  37: '37°C',
  100: {
    style: {
      color: '#f50',
    },
    label: <strong>100°C</strong>,
  },
};

const App: React.FC = () => (
  <div>
    <h4>included=true</h4>
    <Slider marks={marks} defaultValue={37} />
    <Slider range marks={marks} defaultValue=100 />
  </div>
  <div>
    <h4>included=false</h4>
    <Slider marks={marks} included={false} defaultValue=37 />
  </div>
  <div>
    <h4>marks & step</h4>
    <Slider marks={marks} step={10} defaultValue=30 />
  </div>
  <div>
    <h4>step=null</h4>
    <Slider marks={marks} step={null} defaultValue=30 />
  </div>
);

export default App;
```



Reversed: ☐

Reverse [✎](#)


Using `reverse` to render slider reversely.

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

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

  return (
    <div>
      <Slider defaultValue={30} reverse={reverse}>
      <Slider range defaultValue={[20, 50]} reversed={reverse}>
      Reversed: <Switch size="small" checked={reverse}>
    </div>
  );
};

export default App;
```




Control visible of Tooltip [✎](#)

When `tooltip.open` is `true`, Tooltip will always show, or Tooltip will not show anyway, even if dragging or hovering.

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

const App: React.FC = () => <Slider defaultValue=
```

export default App;



Draggable track [✎](#)

Make range track draggable when set `range.draggableTrack`.

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

const App: React.FC = () => <Slider range={{ c
```

export default App;

API

Property	Description	Type	Default	Version
autoFocus	Whether get focus when component mounted	boolean	false	
defaultValue	The default value of slider. When <code>range</code> is false, use number, otherwise, use <code>[number, number]</code>	number [number, number]	0 [0, 0]	
disabled	If true, the slider will not be intractable	boolean	false	
keyboard	Support using keyboard to move handlers	boolean	true	5.2.0+
dots	Whether the thumb can drag over tick only	boolean	false	

Property	Description	Type	Default	Version
included	Make effect when <code>marks</code> not null, true means containment and false means coordinative	boolean	true	
marks	Tick mark of Slider, type of key must be <code>number</code> , and must in closed interval [min, max], each mark can declare its own style	object	<pre>{ number: ReactNode } { number: { style: CSSProperties, label: ReactNode } }</pre>	
max	The maximum value the slider can slide to	number	100	
min	The minimum value the slider can slide to	number	0	
range	Dual thumb mode	boolean	false	
reverse	Reverse the component	boolean	false	
step	The granularity the slider can step through values. Must greater than 0, and be divided by (max - min) . When <code>marks</code> no null, <code>step</code> can be null	number null	1	
tooltip	The tooltip relate props	tooltip	-	4.23.0
value	The value of slider. When <code>range</code> is false, use number, otherwise, use [number, number]	number [number, number]	-	
vertical	If true, the slider will be vertical	boolean	false	
onAfterChange	Fire when onmouseup is fired	(value) => void	-	
onChange	Callback function that is fired when the user changes the slider's value	(value) => void	-	
trackStyle	The style of slider track (the active range)	CSSProperties	-	

Property	Description	Type	Default	Version
railStyle	The style of slider rail (the background)	CSSProperties	-	
handleStyle	The style of slider handle	CSSProperties	-	

range

Property	Description	Type	Default	Version
draggableTrack	Whether range track can be drag	boolean	false	4.10.0

tooltip

Property	Description	Type	Default	Version
open	If true, Tooltip will show always, or it will not show anyway, even if dragging or hovering	boolean	-	4.23.0
placement	Set Tooltip display position. Ref Tooltip	string	-	4.23.0
getPopupContainer	The DOM container of the Tooltip, the default behavior is to create a div element in body	(triggerNode) => HTMLElement	() => document.body	4.23.0
formatter	Slider will pass its value to <code>formatter</code> , and display its value in Tooltip, and hide Tooltip when return value is null	value => ReactNode null	IDENTITY	4.23.0

Methods

Name	Description	Version
blur()	Remove focus	
focus()	Get focus	

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	<code>#ffffff</code>
colorBgElevated	Container background color of the popup layer, in dark mode the color value of this token will be a little brighter than `colorBgContainer`. E.g: modal, pop-up, menu, etc.	string	<code>#ffffff</code>
colorBorderSecondary	Slightly lighter than the default border color, this color is the same as `colorSplit`. Solid color is used.	string	<code>#f0f0f0</code>
colorFillContentHover	Control the style of background color of content area when mouse hovers over it.	string	<code>rgba(0, 0, 0, 0.15)</code>
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	<code>rgba(0, 0, 0, 0.06)</code>
colorFillTertiary	The third level of fill color is used to outline the shape of the element, such as Slider, Segmented, etc. If there is no emphasis requirement, it is recommended to use the third level of fill color as the default fill color.	string	<code>rgba(0, 0, 0, 0.04)</code>
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	<code>#1677ff</code>
colorPrimaryBorder	The stroke color under the main color gradient, used on the stroke of components such as Slider.	string	<code>#91caff</code>
colorPrimaryBorderHover	The hover state of the stroke color under the main color gradient, which will be used when the stroke Hover of components such as Slider and Button.	string	<code>#69b1ff</code>
colorText	Default text color which comply with W3C standards, and this color is also the darkest neutral color.	string	<code>rgba(0, 0, 0, 0.88)</code>
colorTextDescription	Control the font color of text description.	string	<code>rgba(0, 0, 0, 0.45)</code>
colorTextDisabled	Control the color of text in disabled state.	string	<code>rgba(0, 0, 0, 0.25)</code>
borderRadiusXS	XS size border radius, used in some small border radius components, such as Segmented, Arrow and other components with small border radius.	number	2

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
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
lineHeight	Line height of text.	number	1.5714285714285714
lineWidth	Border width of base components	number	1
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