## When To Use

To input a value in a range.

## Examples

### Basic

```tsx
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} />
      Disabled: <Switch size="small" checked={disabled} onChange={onChange}
/>
    </>
  );
};

export default App;
```

### Slider with InputNumber

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

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

  const onChange: InputNumberProps['onChange'] = (newValue) => {
    setInputValue(newValue as number);
  };

  return (
    <Row>
      <Col span={12}>
```

```
        <Slider
          min={1}
          max={20}
          onChange={onChange}
          value={typeof inputValue === 'number' ? inputValue : 0}
        />
      </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);

  const onChange: InputNumberProps['onChange'] = (value) => {
    if (Number.isNaN(value)) {
      return;
    }
    setInputValue(value as number);
  };

  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>
  );
};

const App: React.FC = () => (
  <Space style={{ width: '100%' }} direction="vertical">
    <IntegerStep />
    <DecimalStep />
  </Space>
);

export default App;
```

### Slider with icon

```
import React, { useState } from 'react';
import { FrownOutlined, SmileOutlined } from '@ant-design/icons';
import { Slider } from 'antd';

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

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

  const mid = Number(((max - min) / 2).toFixed(5));
  const preColorCls = value >= mid ? '' : 'icon-wrapper-active';
  const nextColorCls = value >= mid ? 'icon-wrapper-active' : '';

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

```
const App: React.FC = () => <IconSlider min={0} max={20} />;

export default App;
```

**Customize tooltip**

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

const formatter: NonNullable<SliderSingleProps['tooltip']>['formatter'] =
(value) => `${value}%`;

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

export default App;
```

**Event**

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

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

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

const App: React.FC = () => (
  <>
    <Slider defaultValue={30} onChange={onChange} onChangeComplete=
{onChangeComplete} />
    <Slider
      range
      step={10}
      defaultValue={[20, 50]}
      onChange={onChange}
      onChangeComplete={onChangeComplete}
    />
  </>
);

export default App;
```

**Graduated slider**

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

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

const App: React.FC = () => (
  <>
    <h4>included=true</h4>
    <Slider marks={marks} defaultValue={37} />
    <Slider range marks={marks} defaultValue={[26, 37]} />

    <h4>included=false</h4>
    <Slider marks={marks} included={false} defaultValue={37} />

    <h4>marks & step</h4>
    <Slider marks={marks} step={10} defaultValue={37} />

    <h4>step=null</h4>
    <Slider marks={marks} step={null} defaultValue={37} />
  </>
);

export default App;
```

**Vertical**

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

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

const marks: SliderSingleProps['marks'] = {
  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} defaultValue={[20, 50]} />
    </div>
    <div style={style}>
      <Slider vertical range marks={marks} defaultValue={[26, 37]} />
    </div>
  </>
);

export default App;
```

### Control visible of ToolTip

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

const App: React.FC = () => <Slider defaultValue={30} tooltip={{ open: true
}} />;
```

```
export default App;
```

**Reverse**

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

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

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

export default App;
```

**Draggable track**

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

const App: React.FC = () => <Slider range={{ draggableTrack: true }}
defaultValue={[20, 50]} />;

export default App;
```

**Multiple handles**

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

function getGradientColor(percentage: number) {
  const startColor = [135, 208, 104];
  const endColor = [255, 204, 199];

  const midColor = startColor.map((start, i) => {
    const end = endColor[i];
    const delta = end - start;
    return (start + delta * percentage).toFixed(0);
  });

  return `rgb(${midColor.join(',')})`;
}

const App: React.FC = () => {
  const [value, setValue] = React.useState([0, 10, 20]);

  const start = value[0] / 100;
  const end = value[value.length - 1] / 100;

  return (
    <Slider
      range
      defaultValue={value}
      onChange={setValue}
      styles={{
        track: {
          background: 'transparent',
        },
        tracks: {
          background: `linear-gradient(to right, ${getGradientColor(start)}
0%, ${getGradientColor(
            end,
          )} 100%)`,
        },
      }}
    />
  );
};

export default App;
```

### Dynamic edit nodes

v5.20.0

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

const App: React.FC = () => {
  const [value, setValue] = React.useState([20, 80]);

  return (
    <Slider
      range={{ editable: true, minCount: 1, maxCount: 5 }}
      value={value}
      onChange={setValue}
    />
  );
};

export default App;
```

### Component Token

Debug

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

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

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

const App: React.FC = () => (
  <ConfigProvider
    theme={{
      components: {
        Slider: {
          controlSize: 20,
          railSize: 4,
          handleSize: 22,
          handleSizeHover: 18,
          dotSize: 8,
          handleLineWidth: 6,
          handleLineWidthHover: 2,
          railBg: '#9f3434',
          railHoverBg: '#8d2424',
          trackBg: '#b0b0ef',
          trackHoverBg: '#c77195',
          handleColor: '#e6f6a2',
          handleActiveColor: '#d22bc4',
          dotBorderColor: '#303030',
          dotActiveBorderColor: '#918542',
          trackBgDisabled: '#1a1b80',
        },
      },
    }}
  >
    <Slider defaultValue={30} disabled />
    <Slider range={{ draggableTrack: true }} defaultValue={[20, 50]} />
    <div style={style}>
      <Slider vertical defaultValue={30} />
    </div>
    <div style={style}>
      <Slider vertical range step={10} defaultValue={[20, 50]} />
    </div>
    <div style={style}>
      <Slider vertical range marks={marks} defaultValue={[26, 37]} />
    </div>
  </ConfigProvider>
);

export default App;
```

## API

Common props ref： [Common props]

| Property | Description | Type | Default | Version |
|---|---|---|---|---|
| autoFocus | Whether get focus when component mounted | boolean | false | |
| classNames | Semantic structure className | [Record<SemanticDOM, string>] | - | 5.10.0 |
| defaultValue | The default value of slider. When range is | number \| [number, number] | 0 \| [0, 0] | |

| | | | | |
|---|---|---|---|---|
| | false, use number, otherwise, use [number, number] | | | |
| 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 | |
| included | Make effect when `marks` not null, true means containment and false means coordinative | boolean | true | |
| marks | Tick mark of Slider, type of key must be `number`, and must in closed interval [min, max], each mark can declare its own style | object | { number: ReactNode } \| { number: { style: CSSProperties, label: ReactNode } } | |
| max | The maximum value the | number | 100 | |

| | | | | |
|---|---|---|---|---|
| | slider can slide to | | | |
| 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 `step` is `null` but exist `marks`, the valid point will only be the `mark`, `min` and `max` | number \| null | 1 | |
| styles | Semantic structure style | [Record<SemanticDOM, React.CSSProperties>]() | - | 5.10.0 |
| tooltip | The tooltip relate props | [tooltip]() | - | 4.23.0 |
| value | The value of slider. When `range` is false, use number, otherwise, use | number \| [number, number] | - | |

| | | | | |
|---|---|---|---|---|
| | [number, number] | | | |
| vertical | If true, the slider will be vertical | boolean | false | |
| onChangeComplete | Fire when `mouseup` or `keyup` is fired | (value) => void | - | |
| onChange | Callback function that is fired when the user changes the slider's value | (value) => void | - | |

### range

| Property | Description | Type | Default | Version |
|---|---|---|---|---|
| draggableTrack | Whether range track can be drag | boolean | false | - |
| editable | Dynamic edit nodes, can't be used with `draggableTrack` | boolean | false | 5.20.0 |
| minCount | The minimum count of nodes | number | 0 | 5.20.0 |
| maxCount | The maximum count of nodes | number | - | 5.20.0 |

### tooltip

| Property | Description | Type | Default | Version |
|---|---|---|---|---|
| autoAdjustOverflow | Whether to automatically adjust the popup position | boolean | true | 5.8.0 |
| 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 `formatter`, 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 |  |

## Semantic DOM

**演示**

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

import SemanticPreview from '../../../.dumi/components/SemanticPreview';
import useLocale from '../../../.dumi/hooks/useLocale';

const locales = {
  cn: {
    root: '根元素',
    track: '范围选择下，点和点之间单个选取条',
    tracks: '范围选择下，整个范围选取条',
    rail: '背景条元素',
    handle: '抓取点元素',
  },
  en: {
    root: 'Root element',
    track: 'The selection bar between points and points under the range
selection',
    tracks: 'The entire range selection bar under the range selection',
    rail: 'Background rail element',
    handle: 'Grab handle element',
  },
};

const App: React.FC = () => {
  const [locale] = useLocale(locales);
  return (
    <SemanticPreview
      componentName="Slider"
      semantics={[
        { name: 'root', desc: locale.root, version: '5.23.0' },
        { name: 'track', desc: locale.track, version: '5.10.0' },
        { name: 'tracks', desc: locale.tracks, version: '5.10.0' },
        { name: 'rail', desc: locale.rail, version: '5.10.0' },
        { name: 'handle', desc: locale.handle, version: '5.10.0' },
      ]}
    >
      <Slider range defaultValue={[20, 30, 50]} style={{ width: '100%' }}
/>
    </SemanticPreview>
  );
};

export default App;
```

## Design Token