何时使用 {#when-to-use}

当用户需要在数值区间/自定义区间内进行选择时，可为连续或离散值。

代码演示

基本

```
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;
```

带输入框的滑块

```
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;
```

带 icon 的滑块

```
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>
  );
};
```

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

export default App;
```

自定义提示

```tsx
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;
```

事件

```tsx
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;
```

带标签的滑块

```
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;
```

垂直

```tsx
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;
```

控制 ToolTip 的显示

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

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

```
export default App;
```

反向

```
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;
```

范围可拖拽

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

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

export default App;
```

多点组合

```
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;
```

动态增减节点

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;
```

组件 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

通用属性参考：通用属性

参数	说明	类型	默认值	版本
autoFocus	自动获取焦点	boolean	false	
classNames	语义化结构 className	Record<SemanticDOM, string>	-	5.10.0
defaultValue	设置初始取值。当 range 为 false 时，使用 number，否则用 [number, number]	number | [number, number]	0 | [0, 0]	

disabled	值为 true 时，滑块为禁用状态	boolean	false	
keyboard	支持使用键盘操作 handler	boolean	true	5.2.0+
dots	是否只能拖拽到刻度上	boolean	false	
included	`marks` 不为空对象时有效，值为 true 时表示值为包含关系，false 表示并列	boolean	true	
marks	刻度标记，key 的类型必须为 `number` 且取值在闭区间 [min, max] 内，每个标签可以单独设置样式	object	{ number: ReactNode } or { number: { style: CSSProperties, label: ReactNode } }	
max	最大值	number	100	
min	最小值	number	0	
range	双滑块模式	boolean | range	false	
reverse	反向坐标轴	boolean	false	
step	步长，取值必须大于 0，并且可被 (max - min) 整除。当 `marks` 不为空对象时，可以设置 `step` 为 null，此时 Slider 的可选值仅有 `marks`、`min` 和 `max`	number | null	1	

styles	语义化结构 styles	Record<SemanticDOM, React.CSSProperties>	-	5.10.0
tooltip	设置 Tooltip 相关属性	tooltip	-	4.23.0
value	设置当前取值。当 `range` 为 false 时，使用 number，否则用 [number, number]	number | [number, number]	-	
vertical	值为 true 时，Slider 为垂直方向	boolean	false	
onChangeComplete	与 `mouseup` 和 keyup 触发时机一致，把当前值作为参数传入	(value) => void	-	
onChange	当 Slider 的值发生改变时，会触发 onChange 事件，并把改变后的值作为参数传入	(value) => void	-	

range

参数	说明	类型	默认值	版本
draggableTrack	范围刻度是否可被拖拽	boolean	false	
editable	启动动态增减节点，不能和 `draggableTrack` 一同使用	boolean	false	5.20.0
minCount	配置 `editable` 时，最小节点数量	number	0	5.20.0
maxCount	配置 `editable` 时，最大节点数量	number	-	5.20.0

tooltip

参数	说明	类型	默认值	版本
autoAdjustOverflow	是否自动调整弹出位置	boolean	true	5.8.0
open	值为 true 时，Tooltip 将会始终显示；否则始终不显示，哪怕在拖拽及移入时	boolean	-	4.23.0
placement	设置 Tooltip 展示位置。参考 Tooltip	string	-	4.23.0
getPopupContainer	Tooltip 渲染父节点，默认渲染到 body 上	(triggerNode) => HTMLElement	() => document.body	4.23.0
formatter	Slider 会把当前值传给 `formatter`，并在 Tooltip 中显示 `formatter` 的返回值，若为 null，则隐藏 Tooltip	value => ReactNode | null	IDENTITY	4.23.0

方法

名称	描述	版本
blur()	移除焦点	
focus()	获取焦点	

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;
```

主题变量（Design Token）