何时使用 {#when-to-use}

当需要获取标准数值时。

代码演示

基本

```
import React from 'react';
import type { InputNumberProps } from 'antd';
import { InputNumber } from 'antd';

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

const App: React.FC = () => <InputNumber min={1} max={10} defaultValue={3}
onChange={onChange} />;

export default App;
```

三种大小

```
import React from 'react';
import type { InputNumberProps } from 'antd';
import { InputNumber, Space } from 'antd';

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

const App: React.FC = () => (
  <Space wrap>
    <InputNumber size="large" min={1} max={100000} defaultValue={3}
onChange={onChange} />
    <InputNumber min={1} max={100000} defaultValue={3} onChange={onChange}
/>
    <InputNumber size="small" min={1} max={100000} defaultValue={3}
onChange={onChange} />
  </Space>
);

export default App;
```

前置/后置标签

```tsx
import React from 'react';
import { SettingOutlined } from '@ant-design/icons';
import { Cascader, InputNumber, Select, Space } from 'antd';

const { Option } = Select;

const selectBefore = (
  <Select defaultValue="add" style={{ width: 60 }}>
    <Option value="add">+</Option>
    <Option value="minus">-</Option>
  </Select>
);
const selectAfter = (
  <Select defaultValue="USD" style={{ width: 60 }}>
    <Option value="USD">$</Option>
    <Option value="EUR">€</Option>
    <Option value="GBP">£</Option>
    <Option value="CNY">¥</Option>
  </Select>
);

const App: React.FC = () => (
  <Space direction="vertical">
    <InputNumber addonBefore="+" addonAfter="$" defaultValue={100} />
    <InputNumber addonBefore={selectBefore} addonAfter={selectAfter}
defaultValue={100} />
    <InputNumber addonAfter={<SettingOutlined />} defaultValue={100} />
    <InputNumber
      addonBefore={<Cascader placeholder="cascader" style={{ width: 150 }}
/>}
      defaultValue={100}
    />
    <InputNumber
      addonBefore="+"
      addonAfter={<SettingOutlined />}
      defaultValue={100}
      disabled
      controls
    />
    <InputNumber
      prefix="¥"
      addonBefore="+"
      addonAfter={<SettingOutlined />}
      defaultValue={100}
      disabled
      controls
```

```
    />
  </Space>
);

export default App;
```

不可用

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

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

  const toggle = () => {
    setDisabled(!disabled);
  };

  return (
    <>
      <InputNumber min={1} max={10} disabled={disabled} defaultValue={3} />
      <div style={{ marginTop: 20 }}>
        <Button onClick={toggle} type="primary">
          Toggle disabled
        </Button>
      </div>
    </>
  );
};

export default App;
```

高精度小数

```
import React from 'react';
import type { InputNumberProps } from 'antd';
import { InputNumber } from 'antd';

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

const App: React.FC = () => (
  <InputNumber<string>
    style={{ width: 200 }}
```

```
    defaultValue="1"
    min="0"
    max="10"
    step="0.00000000000001"
    onChange={onChange}
    stringMode
  />
);

export default App;
```

格式化展示

```
import React from 'react';
import type { InputNumberProps } from 'antd';
import { InputNumber, Space } from 'antd';

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

const App: React.FC = () => (
  <Space>
    <InputNumber<number>
      defaultValue={1000}
      formatter={(value) => `$ ${value}`.replace(/\B(?=(\d{3})+(?!\d))/g,
',')}
      parser={(value) => value?.replace(/\$\s?|(,*)/g, '') as unknown as
number}
      onChange={onChange}
    />
    <InputNumber<number>
      defaultValue={100}
      min={0}
      max={100}
      formatter={(value) => `${value}%`}
      parser={(value) => value?.replace('%', '') as unknown as number}
      onChange={onChange}
    />
  </Space>
);

export default App;
```

键盘行为

```tsx
import React, { useState } from 'react';
import { Checkbox, InputNumber, Space } from 'antd';

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

  return (
    <Space>
      <InputNumber min={1} max={10} keyboard={keyboard} defaultValue={3} />
      <Checkbox
        onChange={() => {
          setKeyboard(!keyboard);
        }}
        checked={keyboard}
      >
        Toggle keyboard
      </Checkbox>
    </Space>
  );
};

export default App;
```

鼠标滚轮

v5.14.0

```tsx
import React from 'react';
import type { InputNumberProps } from 'antd';
import { InputNumber } from 'antd';

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

const App: React.FC = () => (
  <InputNumber min={1} max={10} defaultValue={3} onChange={onChange}
changeOnWheel />
);

export default App;
```

形态变体

v5.13.0

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

const App: React.FC = () => (
  <Flex vertical gap={12}>
    <InputNumber placeholder="Outlined" style={{ width: 200 }} />
    <InputNumber placeholder="Filled" variant="filled" style={{ width: 200
}} />
    <InputNumber placeholder="Borderless" variant="borderless" style={{
width: 200 }} />
    <InputNumber placeholder="Underlined" variant="underlined" style={{
width: 200 }} />
  </Flex>
);

export default App;
```

Filled Debug

Debug

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

const App: React.FC = () => (
  <Flex vertical gap={12}>
    <Flex gap={12}>
      <InputNumber placeholder="Filled" variant="filled" />
      <InputNumber placeholder="Filled" variant="filled" disabled />
      <InputNumber placeholder="Filled" variant="filled" status="error" />
    </Flex>
    <Flex gap={12}>
      <InputNumber prefix="$" placeholder="Filled" variant="filled" />
      <InputNumber prefix="$" placeholder="Filled" variant="filled"
disabled />
      <InputNumber prefix="$" placeholder="Filled" variant="filled"
status="error" />
    </Flex>
    <Flex gap={12}>
      <InputNumber addonBefore="http://" addonAfter=".com"
placeholder="Filled" variant="filled" />
      <InputNumber
        addonBefore="http://"
        addonAfter=".com"
        placeholder="Filled"
        variant="filled"
```

```
        disabled
      />
      <InputNumber
        addonBefore="http://"
        addonAfter=".com"
        placeholder="Filled"
        variant="filled"
        status="error"
      />
    </Flex>
    <Flex gap={12}>
      <InputNumber addonAfter=".com" placeholder="Filled" variant="filled"
/>
      <InputNumber addonAfter=".com" placeholder="Filled" variant="filled"
disabled />
      <InputNumber addonAfter=".com" placeholder="Filled" variant="filled"
status="error" />
    </Flex>
    <Flex gap={12}>
      <InputNumber addonBefore="http://" placeholder="Filled"
variant="filled" />
      <InputNumber addonBefore="http://" placeholder="Filled"
variant="filled" disabled />
      <InputNumber addonBefore="http://" placeholder="Filled"
variant="filled" status="error" />
    </Flex>
  </Flex>
);

export default App;
```

超出边界

```
import React, { useState } from 'react';
import { Button, InputNumber, Space } from 'antd';

const App: React.FC = () => {
  const [value, setValue] = useState<string | number | null>('99');

  return (
    <Space>
      <InputNumber min={1} max={10} value={value} onChange={setValue} />
      <Button
        type="primary"
        onClick={() => {
          setValue(99);
```

```
        }}
      >
        Reset
      </Button>
    </Space>
  );
};

export default App;
```

前缀/后缀

```
import React from 'react';
import { UserOutlined } from '@ant-design/icons';
import { InputNumber } from 'antd';

const App: React.FC = () => (
  <>
    <InputNumber prefix="￥" style={{ width: '100%' }} />
    <br />
    <br />
    <InputNumber addonBefore={<UserOutlined />} prefix="￥" style={{ width:
'100%' }} />
    <br />
    <br />
    <InputNumber prefix="￥" disabled style={{ width: '100%' }} />
    <br />
    <br />
    <InputNumber suffix="RMB" style={{ width: '100%' }} />
  </>
);

export default App;
```

自定义状态

```
import React from 'react';
import ClockCircleOutlined from '@ant-design/icons/ClockCircleOutlined';
import { InputNumber, Space } from 'antd';

const App: React.FC = () => (
  <Space direction="vertical" style={{ width: '100%' }}>
    <InputNumber status="error" style={{ width: '100%' }} />
    <InputNumber status="warning" style={{ width: '100%' }} />
    <InputNumber status="error" style={{ width: '100%' }} prefix=
```

```
{<ClockCircleOutlined />} />
    <InputNumber status="warning" style={{ width: '100%' }} prefix=
{<ClockCircleOutlined />} />
  </Space>
);

export default App;
```

聚焦

v5.22.0

```
import React, { useRef } from 'react';
import { Button, InputNumber, Space } from 'antd';
import type { InputNumberRef } from 'rc-input-number';

const App: React.FC = () => {
  const inputRef = useRef<InputNumberRef>(null);

  return (
    <Space direction="vertical" style={{ width: '100%' }}>
      <Space wrap>
        <Button
          onClick={() => {
            inputRef.current!.focus({
              cursor: 'start',
            });
          }}
        >
          Focus at first
        </Button>
        <Button
          onClick={() => {
            inputRef.current!.focus({
              cursor: 'end',
            });
          }}
        >
          Focus at last
        </Button>
        <Button
          onClick={() => {
            inputRef.current!.focus({
              cursor: 'all',
            });
          }}
```

```
        >
          Focus to select all
        </Button>
        <Button
          onClick={() => {
            inputRef.current!.focus({
              preventScroll: true,
            });
          }}
        >
          Focus prevent scroll
        </Button>
      </Space>
      <InputNumber style={{ width: '100%' }} defaultValue={999} ref=
{inputRef} />
    </Space>
  );
};

export default App;
```

图标按钮

Debug

```
import React from 'react';
import { ArrowDownOutlined, ArrowUpOutlined } from '@ant-design/icons';
import { InputNumber } from 'antd';

const App: React.FC = () => (
  <InputNumber controls={{ upIcon: <ArrowUpOutlined />, downIcon:
<ArrowDownOutlined /> }} />
);

export default App;
```

_InternalPanelDoNotUseOrYouWillBeFired

Debug

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

/** Test usage. Do not use in your production. */
const { _InternalPanelDoNotUseOrYouWillBeFired: InternalInputNumber } =
InputNumber;
```

```tsx
export default () => (
  <div style={{ display: 'flex', flexDirection: 'column', rowGap: 16 }}>
    <InternalInputNumber style={{ width: '100%' }} />
  </div>
);
```

覆盖组件样式

Debug

```tsx
import React from 'react';
import { ConfigProvider, InputNumber, Space } from 'antd';

export default () => (
  <ConfigProvider
    theme={{
      components: {
        InputNumber: {
          handleWidth: 50,
        },
      },
    }}
  >
    <Space wrap>
      <InputNumber />

      <ConfigProvider
        theme={{
          components: {
            InputNumber: {
              handleWidth: 25,
            },
          },
        }}
      >
        <InputNumber />
      </ConfigProvider>

      <ConfigProvider
        theme={{
          components: {
            InputNumber: {
              paddingBlockLG: 12,
              paddingInlineLG: 16,
            },
```

```
          },
        }}
      >
        <Space wrap>
          <InputNumber size="large" />
          <InputNumber size="large" prefix="$" />
        </Space>
      </ConfigProvider>

      <ConfigProvider
        theme={{
          components: {
            InputNumber: {
              inputFontSize: 30,
              inputFontSizeSM: 20,
              inputFontSizeLG: 40,
            },
          },
        }}
      >
        <Space wrap>
          <InputNumber defaultValue={11111} size="small" />
          <InputNumber defaultValue={11111} />
          <InputNumber defaultValue={11111} size="large" />
        </Space>
      </ConfigProvider>
    </Space>
  </ConfigProvider>
);
```

API

通用属性参考：通用属性

参数	说明	类型	默认值	
addonAfter	带标签的 input，设置后置标签	ReactNode	-	4.17.
addonBefore	带标签的 input，设置前置标签	ReactNode	-	4.17.
autoFocus	自动获取焦点	boolean	false	-

changeOnBlur	是否在失去焦点时，触发 onChange 事件（例如值超出范围时，重新限制回范围并触发事件）	boolean	true	5.11.
changeOnWheel	允许鼠标滚轮改变数值	boolean	-	5.14.
controls	是否显示增减按钮，也可设置自定义箭头图标	boolean | { upIcon?: React.ReactNode; downIcon?: React.ReactNode; }	-	4.19.
decimalSeparator	小数点	string	-	-
placeholder	占位符	string	-	
defaultValue	初始值	number	-	-
disabled	禁用	boolean	false	-
formatter	指定输入框展示值的格式	function(value: number | string, info: { userTyping: boolean, input: string }): string	-	info:
keyboard	是否启用键盘快捷行为	boolean	true	4.12.
max	最大值	number	Number.MAX_SAFE_INTEGER	-
min	最小值	number	Number.MIN_SAFE_INTEGER	-
parser	指定从 `formatter` 里转换回数字的方式，和 `formatter` 搭配使用	function(string): number	-	-

precision	数值精度，配置 `formatter` 时会以 `formatter` 为准	number	-	-
readOnly	只读	boolean	false	-
status	设置校验状态	'error' | 'warning'	-	4.19.
prefix	带有前缀图标的 input	ReactNode	-	4.17.
suffix	带有后缀图标的 input	ReactNode	-	5.20
size	输入框大小	`large` | `middle` | `small`	-	-
step	每次改变步数，可以为小数	number | string	1	-
stringMode	字符值模式，开启后支持高精度小数。同时 `onChange` 将返回 string 类型	boolean	false	4.13.
value	当前值	number	-	-
variant	形态变体	`outlined` | `borderless` | `filled` | `underlined`	`outlined`	5.13. unde 5.24
onChange	变化回调	function(value: number | string | null)	-	-
onPressEnter	按下回车的回调	function(e)	-	-
onStep	点击上下箭头的回调	(value: number, info: { offset: number, type:	-	4.7.0

		'up' | 'down' }) => void		

Ref

名称	说明	参数	版本
blur()	移除焦点	-	
focus()	获取焦点	(option?: { preventScroll?: boolean, cursor?: 'start' | 'end' | 'all' })	cursor - 5.22.0
nativeElement	获取原生 DOM 元素	-	5.17.3

主题变量（Design Token）

FAQ

为何受控模式下，`value` 可以超出 `min` 和 `max` 范围?

在受控模式下，开发者可能自行存储相关数据。如果组件将数据约束回范围内，会导致展示数据与实际存储数据不一致的情况。这使得一些如表单场景存在潜在的数据问题。

为何动态修改 `min` 和 `max` 让 `value` 超出范围不会触发 `onChange` 事件?

`onChange` 事件为用户触发事件，自行触发会导致表单库误以为变更来自用户操作。我们以错误样式展示超出范围的数值。

为何 `onBlur` 等事件获取不到正确的 value?

InputNumber 的值由内部逻辑封装而成，通过 `onBlur` 等事件获取的 `event.target.value` 仅为 DOM 元素的 `value` 而非 InputNumber 的实际值。例如通过 `formatter` 或者 `decimalSeparator` 更改展示格式，DOM 中得到的就是格式化后的字符串。你总是应该通过 `onChange` 获取当前值。

为何 `changeOnWheel` 无法控制鼠标滚轮是否改变数值?

> *不建议使用 `type` 属性*

InputNumber 组件允许你使用 input 元素的所有属性最终透传至 input 元素，当你传入 `type="number"` 时 input 元素也会添加这个属性，这会使 input 元素触发原生特性（允许鼠标滚轮改变数值），从而导致 `changeOnWheel` 无法控制鼠标滚轮是否改变数值。