When To Use

When a numeric value needs to be provided.

Examples

Basic

```
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;
```

Sizes

```
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;
```

Pre / Post tab

```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;
```

Disabled

```
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;
```

High precision decimals

```
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;
```

Formatter

```
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;
```

Keyboard

```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;
```

Wheel

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;
```

Variants

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;
```

Out of range

```
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;
```

Prefix / Suffix

```
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;
```

Status

```
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;
```

Focus

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;
```

Icon

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;
```

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

Override Component Style

Debug

```
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

Common props ref：Common props

Property	Description	Type	Default	V
addonAfter	The label text displayed after (on the right side of) the input field	ReactNode	-	
addonBefore	The label text displayed	ReactNode	-	

	before (on the left side of) the input field			
autoFocus	If get focus when component mounted	boolean	false	-
changeOnBlur	Trigger onChange when blur. e.g. reset value in range by blur	boolean	true	5.11
changeOnWheel	Allow control with mouse wheel	boolean	-	5.14
controls	Whether to show +– controls, or set custom arrows icon	boolean | { upIcon?: React.ReactNode; downIcon?: React.ReactNode; }	-	4.19
decimalSeparator	Decimal separator	string	-	-
placeholder	placeholder	string	-	
defaultValue	The initial value	number	-	-
disabled	If disable the input	boolean	false	-
formatter	Specifies the format of the value presented	function(value: number | string, info: { userTyping: boolean, input: string }): string	-	info
keyboard	If enable keyboard behavior	boolean	true	4.12

max	The max value	number	Number.MAX_SAFE_INTEGER	-
min	The min value	number	Number.MIN_SAFE_INTEGER	-
parser	Specifies the value extracted from formatter	function(string): number	-	-
precision	The precision of input value. Will use `formatter` when config of `formatter`	number	-	-
readOnly	If readonly the input	boolean	false	-
status	Set validation status	'error' | 'warning'	-	4.19
prefix	The prefix icon for the Input	ReactNode	-	4.17
suffix	The suffix icon for the Input	ReactNode	-	5.20
size	The height of input box	`large` | `middle` | `small`	-	-
step	The number to which the current value is increased or decreased. It can be an integer or decimal	number | string	1	-

stringMode	Set value as string to support high precision decimals. Will return string value by onChange	boolean	false	4.13
value	The current value	number	-	-
variant	Variants of Input	`outlined` | `borderless` | `filled` | `underlined`	`outlined`	5.13 und 5.24
onChange	The callback triggered when the value is changed	function(value: number | string | null)	-	-
onPressEnter	The callback function that is triggered when Enter key is pressed	function(e)	-	-
onStep	The callback function that is triggered when click up or down buttons	(value: number, info: { offset: number, type: 'up' | 'down' }) => void	-	4.7.(

Ref

Name	Description	Type	Version

blur()	Remove focus	-	
focus()	Get focus	(option?: { preventScroll?: boolean, cursor?: 'start' | 'end' | 'all' })	cursor - 5.22.0
nativeElement	The native DOM element	-	5.17.3

Design Token

Notes

Per issues #21158, #17344, #9421, and documentation about inputs, it appears this community does not support native inclusion of the `type="number"` in the `<Input />` attributes, so please feel free to include it as needed, and be aware that it is heavily suggested that server side validation be utilized, as client side validation can be edited by power users.

FAQ

Why `value` can exceed `min` or `max` in control?

Developer handle data by their own in control. It will make data out of sync if InputNumber change display value. It also cause potential data issues when use in form.

Why dynamic change `min` or `max` which makes `value` out of range will not trigger `onChange` ?

`onChange` is user trigger event. Auto trigger will makes form lib can not detect data modify source.

Why `onBlur` or other event can not get correct value?

InputNumber's value is wrapped by internal logic. The `event.target.value` you get from `onBlur` or other event is the DOM element's `value` instead of the actual value of InputNumber. For example, if you change the display format through `formatter` or `decimalSeparator`, you will get the formatted string in the DOM. You should always get the current value through `onChange`.

Why `changeOnWheel` unable to control whether the mouse scroll wheel changes value?

> *The use of the `type` attribute is deprecated*

The InputNumber component allows you to use all the attributes of the input element and ultimately pass them to the input element, This attribute will also be added to the input element when you pass in `type='number'`, which will cause the input element to trigger native properties (allowing the mouse wheel to change the value), As a result `changeOnWheel` cannot control whether the mouse wheel changes the value.