

# Input

A basic widget for getting the user input is a text field. Keyboard and mouse can be used for providing or changing data.

## When To Use

- A user input in a form field is needed.
- A search input is required.

## Examples

## Basic usage

### Basic usage [↗](#)

Basic usage example.

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

const App: React.FC = () => <Input placeholder="Basic usage" />;

export default App;
```

### Pre / Post tab [↗](#)

Using pre & post tabs example.

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

const { Option } = Select;

const selectBefore = (
  <Select defaultValue="http://">
    <Option value="http://">http://</Option>
    <Option value="https://">https://</Option>
  </Select>
);

const selectAfter = (
  <Select defaultValue=".com">
    <Option value=".com">.com</Option>
    <Option value=".jp">.jp</Option>
    <Option value=".cn">.cn</Option>
    <Option value=".org">.org</Option>
  </Select>
);

const App: React.FC = () => (
  <Space direction="vertical">
    <Input addonBefore="http://" addonAfter="http://" />
    <Input addonBefore={selectBefore} addonAfter={selectAfter} />
    <Input addonAfter={<SettingOutlined />} />
    <Input addonBefore="http://" suffix=".com" />
    <Input
      addonBefore={<Cascader placeholder="cascader" />
      defaultValue="mysite"
    />
  </Space>
);

export default App;
```

## large size

## default size

## small size

### Three sizes of Input [↗](#)

There are three sizes of an Input box: **large** (40px), **default** (32px) and **small** (24px).

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

const App: React.FC = () => (
  <>
    <Input size="large" placeholder="large size" />
    <br />
    <Input placeholder="default size" prefix={<UserOutlined />} />
    <br />
    <Input size="small" placeholder="small size" />
  </>
);

export default App;
```

### Search box [🔗](#)

Example of creating a search box by grouping a standard input with a search button.

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

const { Search } = Input;

const suffix = (
  <AudioOutlined
    style={{
      fontSize: 16,
      color: '#1890ff',
    }}
  />
);

const onSearch = (value: string) => console.log(value);

const App: React.FC = () => (
  <Space direction="vertical">
    <Search placeholder="input search text" or
    <Search placeholder="input search text" al
    <Search
      addonBefore="https://"
      placeholder="input search text"
      allowClear
      onSearch={onSearch}
      style={{ width: 304 }}
    />
    <Search placeholder="input search text" or
    <Search
      placeholder="input search text"
      allowClear
      enterButton="Search"
      size="large"
      onSearch={onSearch}
    />
    <Search
      placeholder="input search text"
      enterButton="Search"
      size="large"
      suffix={suffix}
      onSearch={onSearch}
    />
  </Space>
);

export default App;
```

### Compact Style [🔗](#)

Use `Space.Compact` create compact style, See the [Space.Compact](#) documentation for more.

```
import { SearchOutlined } from '@ant-design/icons';
import { Button, Input, Select, Space } from 'antd';
import React from 'react';

const { Search } = Input;

const options = [
  {
    value: 'zhejiang',
    label: 'Zhejiang',
  },
  {
    value: 'jiangsu',
    label: 'Jiangsu',
  },
];

const App: React.FC = () => (
  <Space direction="vertical" size="middle">
    <Space.Compact>
      <Input defaultValue="26888888" />
    </Space.Compact>
    <Space.Compact>
      <Input style={{ width: '20%' }} defaultValue="26888888" />
      <Input style={{ width: '80%' }} defaultValue="26888888" />
    </Space.Compact>
    <Space.Compact>
      <Search addonBefore="https://" placeholder="input search text" />
    </Space.Compact>
    <Space.Compact style={{ width: '100%' }}>
      <Input defaultValue="Combine input and button" />
      <Button type="primary">Submit</Button>
    </Space.Compact>
    <Space.Compact>
      <Select defaultValue="Zhejiang" options={options} />
      <Input defaultValue="Xihu District, Hangzhou" />
    </Space.Compact>
    <Space.Compact size="large">
      <Input addonBefore={<SearchOutlined />} />
      <Input placeholder="another input" />
    </Space.Compact>
  </Space>
);

export default App;
```

### TextArea

For multi-line input.

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

const { TextArea } = Input;

const App: React.FC = () => (
  <>
    <TextArea rows={4} />
    <br />
    <br />
    <TextArea rows={4} placeholder="maxLength
  </>
);

export default App;
```

### Search box with loading

Search loading when onSearch.

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

const { Search } = Input;

const App: React.FC = () => (
  <>
    <Search placeholder="input search loading
    <br />
    <br />
    <Search placeholder="input search loading
    <br />
    <br />
    <Search placeholder="input search text" er
    </>
  </>
);

export default App;
```

Input a numb...

### Format Tooltip Input

You can use the Input in conjunction with [Tooltip](#) component to create a Numeric Input, which can provide a good experience for extra-long content display.

```
import React, { useState } from 'react';
import { Input, Tooltip } from 'antd';

interface NumericInputProps {
  style: React.CSSProperties;
  value: string;
  onChange: (value: string) => void;
}

const formatNumber = (value: number) => new Intl.NumberFormat().format(value);

const NumericInput = (props: NumericInputProps) => {
  const { value, onChange } = props;

  const handleChange = (e: React.ChangeEvent<HTMLInputElement>) => {
    const { value: inputValue } = e.target;
    const reg = /^-?\d*(\.\d*)?$/;
    if (reg.test(inputValue) || inputValue === '') {
      onChange(inputValue);
    }
  };

  // '.' at the end or only '-' in the input
  const handleBlur = () => {
    let valueTemp = value;
    if (value.charAt(value.length - 1) === '.') {
      valueTemp = value.slice(0, -1);
    }
    onChange(valueTemp.replace(/0*(\d+)/, '$1'));
  };

  const title = value ? (
    <span className="numeric-input-title">{value}</span>
  ) : (
    'Input a number'
  );

  return (
    <Tooltip trigger={['focus']} title={title}>
      <Input
        {...props}
        onChange={handleChange}
        onBlur={handleBlur}
        placeholder="Input a number"
        maxLength={16}
      />
    </Tooltip>
  );
};

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

  return <NumericInput style={{ width: 120 }} value={value} onChange={setValue} />;
};

export default App;
```

Autosize height based on content lines

Autosize height with minimum and maximum number of lines

Controlled autosize

### Autosizing the height to fit the content

`autoSize` prop for a `textarea` type of `Input`

makes the height to automatically adjust based on the content. An option object can be provided to `autoSize` to specify the minimum and maximum number of lines the textarea will automatically adjust.


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

const { TextArea } = Input;


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

  return (
    <div>
      <TextArea placeholder="Autosize height based on content" />
      <div style={{ margin: '24px 0' }} />
      <TextArea
        placeholder="Autosize height with minimum and maximum number of lines"
        autoSize={{ minRows: 2, maxRows: 6 }}
      />
      <div style={{ margin: '24px 0' }} />
      <TextArea
        value={value}
        onChange={(e) => setValue(e.target.value)}
        placeholder="Controlled autosize"
        autoSize={{ minRows: 3, maxRows: 5 }}
      />
    </div>
  );
};

export default App;
```

input password 

input password 

input password  Show

### Password box



Input type of password.

```
import React from 'react';
import { EyeInvisibleOutlined, EyeTwoTone } from 'antd';
import { Button, Input, Space } from 'antd';

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

  return (
    <Space direction="vertical">
      <Input.Password placeholder="input password" type="password" />
      <Input.Password
        placeholder="input password"
        iconRender={(visible) => (visible ? <EyeTwoTone /> : <EyeInvisibleOutlined />)}
        type="password"
      />
      <Space direction="horizontal">
        <Input.Password
          placeholder="input password"
          visibilityToggle={{ visible: passwordVisible }}
          type="password"
        />
        <Button style={{ width: 80 }} onClick={() => setPasswordVisible(!passwordVisible)}>
          {passwordVisible ? 'Hide' : 'Show'}
        </Button>
      </Space>
    </Space>
  );
};

export default App;
```

 Enter your username 

¥ RMB

¥ RMB

### prefix and suffix

Add a prefix or suffix icons inside input.

```
import React from 'react';
import { InfoCircleOutlined, UserOutlined } from 'antd';
import { Input, Tooltip } from 'antd';

const App: React.FC = () => {
  <>
    <Input
      placeholder="Enter your username"
      prefix={<UserOutlined className="site-prefix" />}
      suffix={
        <Tooltip title="Extra information">
          <InfoCircleOutlined style={{ color: 'red' }} />
        </Tooltip>
      }
    />
    <br />
    <br />
    <Input prefix="¥" suffix="RMB" />
    <br />
    <br />
    <Input prefix="¥" suffix="RMB" disabled />
  </>
};

export default App;
```

0 / 20

0 / 100

### With character counting [✎](#)

Show character counting.

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

const { TextArea } = Input;

const onChange = (e: React.ChangeEvent<HTMLInp
  console.log('Change:', e.target.value);
};

const App: React.FC = () => (
  <>
    <Input showCount maxLength={20} onChange={
    <br />
    <br />
    <TextArea showCount maxLength={100} onChar
  </>
);

export default App;
```

input with clear icon

textarea with clear icon

### With clear icon [✎](#)

Input box with the remove icon, click the icon to delete everything.

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

const { TextArea } = Input;

const onChange = (e: React.ChangeEvent<HTMLInp
  console.log(e);
};

const App: React.FC = () => (
  <>
    <Input placeholder="input with clear icon"
    <br />
    <br />
    <TextArea placeholder="textarea with clear
  </>
);

export default App;
```

Error

Warning

 Error with prefix

 Warning with prefix

### Status [✎](#)

Add status to Input with `status`, which could be `error` or `warning`.

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

const App: React.FC = () => (
  <Space direction="vertical" style={{ width:
    <Input status="error" placeholder="Error"
    <Input status="warning" placeholder="Warni
    <Input status="error" prefix={<ClockCircle
    <Input status="warning" prefix={<ClockCirc
  </Space>
);

export default App;
```

Focus at first

Focus at last

Focus to select all

Focus prevent scroll

Input

Ant Design love you!

## Focus

Focus with additional option.

```
import React, { useRef, useState } from 'react'
import type { InputRef } from 'antd';

import { Button, Input, Space, Switch } from 'antd';

const App: React.FC = () => {
  const inputRef = useRef<InputRef>(null);
  const [input, setInput] = useState(true);

  const sharedProps = {
    style: { width: '100%' },
    defaultValue: 'Ant Design love you!',
    ref: inputRef,
  };

  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>
        <Switch
          checked={input}
          checkedChildren="Input"
        />
      </Space>
    </Space>
  );
};
```

can resize

0 / 100

disable resize

0 / 100

## Textarea with character counting

Show character counting.

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

const { TextArea } = Input;

const onChange = (e: React.ChangeEvent<HTMLTextAreaElement>) => {
  console.log('Change:', e.target.value);
};

const App: React.FC = () => {
  <>
    <TextArea
      showCount
      maxLength={100}
      style={{ height: 120, marginBottom: 24 }}
      onChange={onChange}
      placeholder="can resize"
    />
    <TextArea
      showCount
      maxLength={100}
      style={{ height: 120, resize: 'none' }}
      onChange={onChange}
      placeholder="disable resize"
    />
  </>
);

export default App;
```

Borderless

## Borderless

No border.

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

const App: React.FC = () => <Input placeholder="borderless" />

export default App;
```



```

        uncheckedChildren="TextArea"
        onChange={() => {
            setInput(!input);
        }}
    />
</Space>
<br />
{input ? <Input {...sharedProps} /> : <

```

Property	Description	Type	Default	Version
export default App; addonAfter	The label text displayed after (on the right side of) the input field	ReactNode	-	
addonBefore	The label text displayed before (on the left side of) the input field	ReactNode	-	
allowClear	If allow to remove input content with clear icon	boolean   { clearIcon: ReactNode }	false	
bordered	Whether has border style	boolean	true	4.5.0
classNames	Semantic DOM class	Record< <a href="#">SemanticDOM</a> , string>	-	5.4.0
defaultValue	The initial input content	string	-	
disabled	Whether the input is disabled	boolean	false	
id	The ID for input	string	-	
maxLength	The maximum number of characters in Input	number	-	
showCount	Whether to show character count	boolean   { formatter: (info: { value: string, count: number, maxLength?: number ) => ReactNode }	false	4.18.0 info.value: 4.23.0
status	Set validation status	'error'   'warning'	-	4.19.0
styles	Semantic DOM style	Record< <a href="#">SemanticDOM</a> , CSSProperties>	-	5.4.0
prefix	The prefix icon for the Input	ReactNode	-	
size	The size of the input box. Note: in the context of a form, the <code>middle</code> size is used	<code>large</code>   <code>middle</code>   <code>small</code>	-	
suffix	The suffix icon for	ReactNode	-	

Property	Description	Type	Default	Version
	the Input			
type	The type of input, see: <a href="#">MDN</a> ( use <code>Input.TextArea</code> instead of <code>type="textarea" )</code>	string	text	
value	The input content value	string	–	
onChange	Callback when user input	function(e)	–	
onPressEnter	The callback function that is triggered when Enter key is pressed	function(e)	–	

When `Input` is used in a `Form.Item` context, if the `Form.Item` has the `id` and `options` props defined then `value`, `defaultValue`, and `id` props of `Input` are automatically set.

The rest of the props of Input are exactly the same as the original [input](#).

## Input.TextArea

Property	Description	Type	Default	Version
allowClear	If allow to remove input content with clear icon	boolean	false	
autoSize	Height autosize feature, can be set to true   false or an object { minRows: 2, maxRows: 6 }	boolean   object	false	
bordered	Whether has border style	boolean	true	4.5.0
classNames	Semantic DOM class	Record< <a href="#">SemanticDOM</a> , string>	–	5.4.0
defaultValue	The initial input content	string	–	
maxLength	The maximum number of characters in TextArea	number	–	4.7.0
showCount	Whether to show character count	boolean   { formatter: (info: { value: string, count: number, maxLength?: number }) => string }	false	4.7.0 formatter: 4.10.0 info.value: 4.23.0
styles	Semantic DOM style	Record< <a href="#">SemanticDOM</a> , string>	–	5.4.0

Property	Description	Type	Default	Version
CSSProperties>				
value	The input content value	string	–	
onPressEnter	The callback function that is triggered when Enter key is pressed	function(e)	–	
onResize	The callback function that is triggered when resize	function({ width, height })	–	

The rest of the props of `Input.TextArea` are the same as the original [textarea](#).

## Input.Search

Property	Description	Type	Default
enterButton	Whether to show an enter button after input. This property conflicts with the <code>addonAfter</code> property	boolean   ReactNode	false
loading	Search box with loading	boolean	false
onSearch	The callback function triggered when you click on the search-icon, the clear-icon or press the Enter key	function(value, event)	–

Supports all props of `Input`.

## Input.Password

Property	Description	Type	Default
iconRender	Custom toggle button	(visible) => ReactNode	(visible) => (visible ? <EyeOutlined /> : <EyeInvisibleOutlined />)
visibilityToggle	Whether show toggle button or control password visible	boolean   <a href="#">VisibilityToggle</a>	true

## VisibilityToggle

Property	Description	Type	Default	Version
visible	Whether the password is show or hide	boolean	false	4.24.0
onVisibleChange	Callback executed when visibility of the password	boolean	–	4.24.0

Property	Description	Type	Default	Version
	is changed			

Input Methods

Name	Description	Parameters	Version
blur	Remove focus	-	
focus	Get focus	(option?: { preventScroll?: boolean, cursor?: 'start'   'end'   'all' })	option - 4.10.0

Semantic DOM

Input

Property	Description	Version
input	<code>input</code> element	5.4.0
prefix	Wrapper of prefix	5.4.0
suffix	Wrapper of suffix	5.4.0
count	Text count element	5.4.0

Input.TextArea

Property	Description	Version
textarea	<code>textarea</code> element	5.4.0
count	Text count element	5.4.0

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`.	<code>string</code>	<code>#ffffff</code>
colorBgContainerDisabled	Control the background color of container in disabled state.	<code>string</code>	<code>rgba(0, 0, 0, 0.04)</code>
colorBorder	Default border color, used to separate different elements, such as: form separator, card separator, etc.	<code>string</code>	<code>#d9d9d9</code>

Token Name	Description	Type	Default Value
colorError	Used to represent the visual elements of the operation failure, such as the error Button, error Result component, etc.	string	<code>#ff4d4f</code>
colorErrorBorderHover	The hover state border color of the error state.	string	<code>#ffa39e</code>
colorErrorOutline	Control the outline color of input component in error state.	string	<code>rgba(255, 38, 5, 0.06)</code>
colorFillAlter	Control the alternative background color of element.	string	<code>rgba(0, 0, 0, 0.02)</code>
colorIcon	Weak action. Such as `allowClear` or Alert close button	string	<code>rgba(0, 0, 0, 0.45)</code>
colorIconHover	Weak action hover color. Such as `allowClear` or Alert close button	string	<code>rgba(0, 0, 0, 0.88)</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>
colorPrimaryActive	Dark active state under the main color gradient.	string	<code>#0958d9</code>
colorPrimaryHover	Hover state under the main color gradient.	string	<code>#4096ff</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>
colorTextPlaceholder	Control the color of placeholder text.	string	<code>rgba(0, 0, 0, 0.25)</code>
colorTextQuaternary	The fourth level of text color is the lightest text color, such as form input prompt text, disabled color text, etc.	string	<code>rgba(0, 0, 0, 0.25)</code>
colorTextTertiary	The third level of text color is generally used for descriptive text, such as form supplementary explanation text, list descriptive text, etc.	string	<code>rgba(0, 0, 0, 0.45)</code>
colorWarning	Used to represent the warning map token, such as Notification, Alert, etc. Alert or Control component(like Input) will use these map tokens.	string	<code>#faad14</code>

Token Name	Description	Type	Default Value
colorWarningBorderHover	The hover state border color of the warning state.	string	#ffd666
colorWarningOutline	Control the outline color of input component in warning state.	string	rgba(255, 215, 5, 0.1)
borderRadius	Border radius of base components	number	6
borderRadiusLG	LG size border radius, used in some large border radius components, such as Card, Modal and other components.	number	8
borderRadiusSM	SM size border radius, used in small size components, such as Button, Input, Select and other input components in small size	number	4
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
controlOutline	Control the outline color of input component.	string	rgba(5, 145, 255, 0.1)
controlOutlineWidth	Control the outline width of input component.	number	2
controlPaddingHorizontal	Control the horizontal padding of an element.	number	12
controlPaddingHorizontalSM	Control the horizontal padding of an element with a small-medium size.	number	8
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
fontSizeIcon	Control the font size of operation icon in Select, Cascader, etc. Normally same as fontSizeSM.	number	12
fontSizeLG	Large font size	number	16
lineHeight	Line height of text.	number	1.5714285714285714
lineHeightLG	Line height of large text.	number	1.5

Token Name	Description	Type	Default Value
lineType	Border style of base components	string	solid
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
paddingLG	Control the large padding of the element.	number	24
paddingSM	Control the small padding of the element.	number	12
paddingXS	Control the extra small padding of the element.	number	8
paddingXXS	Control the extra extra small padding of the element.	number	4

## FAQ

### Why Input lose focus when change `prefix/suffix/showCount`

When Input dynamic add or remove `prefix/suffix/showCount` will make React recreate the dom structure and new input will be not focused. You can set an empty `<span />` element to keep the dom structure:

```
const suffix = condition ? <Icon type="smile" /> : <span />;

<Input suffix={suffix} />;
```

### Why TextArea in control can make `value` exceed `maxLength` ?

When in control, component should show as what it set to avoid submit value not align with store value in Form.