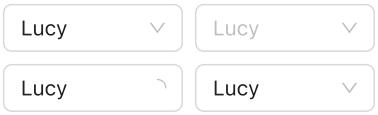
Select

Select component to select value from options.

When To Use

- A dropdown menu for displaying choices - an elegant alternative to the native `<select>` element.
- Utilizing Radio is recommended when there are fewer total options (less than 5).

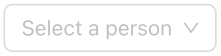
Examples



Basic Usage

Basic Usage.

```
import React from 'react';
import { Select, Space } from 'antd';

const handleChange = (value: string) => {
  console.log(`selected ${value}`);
};

const App: React.FC = () => (
  <Space wrap>
    <Select
      defaultValue="lucy"
      style={{ width: 120 }}
      onChange={handleChange}
      options={[
        { value: 'jack', label: 'Jack' },
        { value: 'lucy', label: 'Lucy' },
        { value: 'Yiminghe', label: 'yiminghe'
        { value: 'disabled', label: 'Disabled'
      ]}
    />
    <Select
      defaultValue="lucy"
      style={{ width: 120 }}
      disabled
      options={[{ value: 'lucy', label: 'Lucy'
    />
    <Select
      defaultValue="lucy"
      style={{ width: 120 }}
      loading
      options={[{ value: 'lucy', label: 'Lucy'
    />
    <Select
      defaultValue="lucy"
      style={{ width: 120 }}
      allowClear
      options={[{ value: 'lucy', label: 'Lucy'
    />
  </Space>
);

export default App;
```



Select with search field

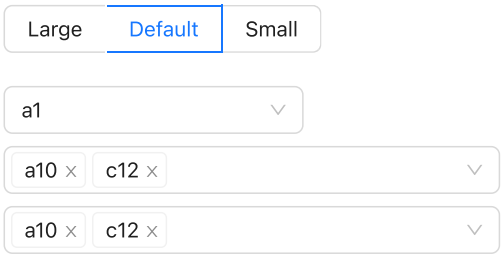
Search the options while expanded.

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

const onChange = (value: string) => {
  console.log(`selected ${value}`);
};

const onSearch = (value: string) => {
  console.log('search:', value);
};

const App: React.FC = () => (
  <Select
    showSearch
    placeholder="Select a person"
    optionFilterProp="children"
    onChange={onChange}
    onSearch={onSearch}
    filterOption={(input, option) =>
      (option?.label ?? '').toLowerCase().incl
    }
    options={[
      {
        value: 'jack',
        label: 'Jack',
      },
      {
        value: 'lucy',
        label: 'Lucy',
      },
      {
        value: 'tom',
        label: 'Tom',
      },
    ]}
  />
);

export default App;
```

multiple selection

Multiple selection, selecting from existing items.

```
import React from 'react';
import { Select, Space } from 'antd';
import type { SelectProps } from 'antd';

const options: SelectProps['options'] = [];

for (let i = 10; i < 36; i++) {
  options.push({
    label: i.toString(36) + i,
    value: i.toString(36) + i,
  });
}

const handleChange = (value: string[]) => {
  console.log(`selected ${value}`);
};

const App: React.FC = () => (
  <Space style={{ width: '100%' }} direction='
    <Select
      mode="multiple"
      allowClear
      style={{ width: '100%' }}
      placeholder="Please select"
      defaultValue={['a10', 'c12']}
      onChange={handleChange}
      options={options}
    />
    <Select
      mode="multiple"
      disabled
      style={{ width: '100%' }}
      placeholder="Please select"
      defaultValue={['a10', 'c12']}
      onChange={handleChange}
      options={options}
    />
  </Space>
);

export default App;
```



Sizes

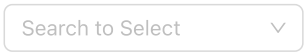
The height of the input field for the select defaults to 32px. If size is set to large, the height will be 40px, and if set to small, 24px.

```
import React, { useState } from 'react';
import { Radio, Select, Space } from 'antd';
import type { SizeType } from 'antd/es/config-
import type { SelectProps, RadioChangeEvent }

const options: SelectProps['options'] = [];

for (let i = 10; i < 36; i++) {
  options.push({
    value: i.toString(36) + i,
    label: i.toString(36) + i,
  });
}

const handleChange = (value: string | string[
  console.log(`Selected: ${value}`);
};

const App: React.FC = () => {
  const [size, setSize] = useState<SizeType>('

  const handleSizeChange = (e: RadioChangeEver
    setSize(e.target.value);
  };

  return (
    <>
      <Radio.Group value={size} onChange={hanc
        <Radio.Button value="large">Large</Rac
        <Radio.Button value="middle">Default</
        <Radio.Button value="small">Small</Rac
      </Radio.Group>
      <br />
      <br />
      <Space direction="vertical" style={{ wic
        <Select
          size={size}
          defaultValue="a1"
          onChange={handleChange}
          style={{ width: 200 }}
          options={options}
        />
        <Select
          mode="multiple"
          size={size}
          placeholder="Please select"
          defaultValue={['a10', 'c12']}
          onChange={handleChange}
          style={{ width: '100%' }}
          options={options}
        />
        <Select
          mode="tags"
          size={size}
```

Custom selection render

Specify the prop name of Option which will be rendered in select box.

```
import React from 'react';
import { Select, Space } from 'antd';

const { Option } = Select;

const handleChange = (value: string[]) => {
  console.log(`selected ${value}`);
};

const App: React.FC = () => (
  <Select
    mode="multiple"
    style={{ width: '100%' }}
    placeholder="select one country"
    defaultValue={['china']}
    onChange={handleChange}
    optionLabelProp="label"
  >
    <Option value="china" label="China">
      <Space>
        <span role="img" aria-label="China">
          🇨🇳
        </span>
        China (中国)
      </Space>
    </Option>
    <Option value="usa" label="USA">
      <Space>
        <span role="img" aria-label="USA">
          🇺🇸
        </span>
        USA (美国)
      </Space>
    </Option>
    <Option value="japan" label="Japan">
      <Space>
        <span role="img" aria-label="Japan">
          🇯🇵
        </span>
        Japan (日本)
      </Space>
    </Option>
    <Option value="korea" label="Korea">
      <Space>
        <span role="img" aria-label="Korea">
          🇰🇷
        </span>
        Korea (韩国)
      </Space>
    </Option>
  </Select>
);

export default App;
```



Search with sort

Search the options with sorting.

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

const App: React.FC = () => (
  <Select
    showSearch
    style={{ width: 200 }}
    placeholder="Search to Select"
    optionFilterProp="children"
    filterOption={(input, option) => (option?.
    filterSort={(optionA, optionB) =>
      (optionA?.label ?? '').toLowerCase().loc
    }
    options={[
      {
        value: '1',
        label: 'Not Identified',
      },
      {
        value: '2',
        label: 'Closed',
      },
      {
        value: '3',
        label: 'Communicated',
      },
      {
        value: '4',
        label: 'Identified',
      },
      {
        value: '5',
        label: 'Resolved',
      },
      {
        value: '6',
        label: 'Cancelled',
      },
    ]}
  />
);

export default App;
```

Tags Mode

Tags

Select with tags, transform input to tag (scroll the menu).

```
import React from 'react';
import { Select } from 'antd';
import type { SelectProps } from 'antd';

const options: SelectProps['options'] = [];

for (let i = 10; i < 36; i++) {
  options.push({
    value: i.toString(36) + i,
    label: i.toString(36) + i,
  });
}

const handleChange = (value: string) => {
  console.log(`selected ${value}`);
};

const App: React.FC = () => (
  <Select
    mode="tags"
    style={{ width: '100%' }}
    placeholder="Tags Mode"
    onChange={handleChange}
    options={options}
  />
);

export default App;
```

Lucy

Option Group

Using `OptGroup` to group the options.

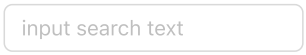
```
import React from 'react';
import { Select } from 'antd';

const handleChange = (value: string) => {
  console.log(`selected ${value}`);
};

const App: React.FC = () => (
  <Select
    defaultValue="lucy"
    style={{ width: 200 }}
    onChange={handleChange}
    options={[
      {
        label: 'Manager',
        options: [
          { label: 'Jack', value: 'jack' },
          { label: 'Lucy', value: 'lucy' },
        ],
      },
      {
        label: 'Engineer',
        options: [{ label: 'yiminghe', value:
      },
    ]}
  />
);

export default App;
```

coordinate

Coordinating the selection of provinces and cities is a common use case and demonstrates how selection can be coordinated.

Using the Cascader component is strongly recommended instead as it is more flexible and capable.

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

const provinceData = ['Zhejiang', 'Jiangsu'];

const cityData = {
  Zhejiang: ['Hangzhou', 'Ningbo', 'Wenzhou'],
  Jiangsu: ['Nanjing', 'Suzhou', 'Zhenjiang'],
};

type CityName = keyof typeof cityData;

const App: React.FC = () => {
  const [cities, setCities] = useState(cityDat
  const [secondCity, setSecondCity] = useState

  const handleProvinceChange = (value: CityNam
    setCities(cityData[value]);
    setSecondCity(cityData[value][0]);
  };

  const onSecondCityChange = (value: CityName)
    setSecondCity(value);
  };

  return (
    <Space wrap>
      <Select
        defaultValue={provinceData[0]}
        style={{ width: 120 }}
        onChange={handleProvinceChange}
        options={provinceData.map((province) =
      />
      <Select
        style={{ width: 120 }}
        value={secondCity}
        onChange={onSecondCityChange}
        options={cities.map((city) => ({ label
      />
    </Space>
  );
};

export default App;
```



Search Box

Search with remote data.

```
import React, { useState } from 'react';
import { Select } from 'antd';
import jsonp from 'fetch-jsonp';
import qs from 'qs';
import type { SelectProps } from 'antd';

let timeout: ReturnType<typeof setTimeout> | r
let currentValue: string;

const fetch = (value: string, callback: Functi
  if (timeout) {
    clearTimeout(timeout);
    timeout = null;
  }
  currentValue = value;

  const fake = () => {
    const str = qs.stringify({
      code: 'utf-8',
      q: value,
    });
    jsonp(`https://suggest.taobao.com/sug?${st
      .then((response: any) => response.json()
      .then((d: any) => {
        if (currentValue === value) {
          const { result } = d;
          const data = result.map((item: any)
            value: item[0],
            text: item[0],
          }));
          callback(data);
        }
      });
  };
  if (value) {
    timeout = setTimeout(fake, 300);
  } else {
    callback([]);
  }
};

const SearchInput: React.FC<{ placeholder: str
  const [data, setData] = useState<SelectProps
  const [value, setValue] = useState<string>()

  const handleSearch = (newValue: string) => {
    fetch(newValue, setData);
  };

  const handleChange = (newValue: string) => {
    setValue(newValue);
  };

  return (
    <Select
      showSearch
      value={value}
      placeholder={props.placeholder}
      style={props.style}
      defaultActiveFirstOption={false}
      showArrow={false}
      filterOption={false}
      onSearch={handleSearch}
      onChange={handleChange}
```

Get value of selected item

As a default behavior, the `onChange` callback can only get the `value` of the selected item. The `labelInValue` prop can be used to get the `label` property of the selected item.

The `label` of the selected item will be packed as an object for passing to the `onChange` callback.

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

const handleChange = (value: { value: string;
  console.log(value); // { value: "lucy", key:
};

const App: React.FC = () => (
  <Select
    labelInValue
    defaultValue={{ value: 'lucy', label: 'Luc
    style={{ width: 120 }}
    onChange={handleChange}
    options={[
      {
        value: 'jack',
        label: 'Jack (100)',
      },
      {
        value: 'lucy',
        label: 'Lucy (101)',
      },
    ]}
  />
);

export default App;
```


Automatic tokenization

Try to copy `Lucy,Jack` and paste to the input.

Only available in tags and multiple mode.

```
import React from 'react';
import { Select } from 'antd';
import type { SelectProps } from 'antd';

const options: SelectProps['options'] = [];

for (let i = 10; i < 36; i++) {
  options.push({
    value: i.toString(36) + i,
    label: i.toString(36) + i,
  });
}

const handleChange = (value: string) => {
  console.log(`selected ${value}`);
};

const App: React.FC = () => (
  <Select
    mode="tags"
    style={{ width: '100%' }}
    onChange={handleChange}
    tokenSeparators={[',']}
    options={options}
  />
);

export default App;
```

Select users

Search and Select Users

A complete multiple select sample with remote search, debounce fetch, ajax callback order flow, and loading state.

```
import React, { useMemo, useRef, useState } fr
import { Select, Spin } from 'antd';
import type { SelectProps } from 'antd/es/sele
import debounce from 'lodash/debounce';

export interface DebounceSelectProps<ValueType
  extends Omit<SelectProps<ValueType | ValueTy
  fetchOptions: (search: string) => Promise<Va
  debounceTimeout?: number;
}

function DebounceSelect<
  ValueType extends { key?: string; label: Rea
>({ fetchOptions, debounceTimeout = 800, ...pr
  const [fetching, setFetching] = useState(fal
  const [options, setOptions] = useState<Value
  const fetchRef = useRef(0);

  const debounceFetcher = useMemo(() => {
    const loadOptions = (value: string) => {
      fetchRef.current += 1;
      const fetchId = fetchRef.current;
      setOptions([]);
      setFetching(true);

      fetchOptions(value).then((newOptions) =>
        if (fetchId !== fetchRef.current) {
          // for fetch callback order
          return;
        }

        setOptions(newOptions);
        setFetching(false);
      });
    };

    return debounce(loadOptions, debounceTimeo
  }, [fetchOptions, debounceTimeout]);

  return (
    <Select
      labelInValue
      filterOption={false}
      onSearch={debounceFetcher}
      notFoundContent={fetching ? <Spin size='
      {...props}
      options={options}
    />
  );
}

// Usage of DebounceSelect
interface UserValue {
  label: string;
  value: string;
}

async function fetchUserList(username: string)
  console.log('fetching user', username);

  return fetch('https://randomuser.me/api/?res
```

custom dropdown render

Custom dropdown

Customize the dropdown menu via `dropdownRender`. If you want to close the dropdown after clicking the custom content, you need to control `open` prop, here is an codesandbox.

```
import React, { useState, useRef } from 'react
import { PlusOutlined } from '@ant-design/icor
import { Divider, Input, Select, Space, Buttor
import type { InputRef } from 'antd';

let index = 0;

const App: React.FC = () => {
  const [items, setItems] = useState(['jack',
  const [name, setName] = useState('');
  const inputRef = useRef<InputRef>(null);

  const onNameChange = (event: React.ChangeEve
    setName(event.target.value);
  };

  const addItem = (e: React.MouseEvent<HTMLBut
    e.preventDefault();
    setItems([...items, name || `New item ${ir
    setName('');
    setTimeout(() => {
      inputRef.current?.focus();
    }, 0);
  };

  return (
    <Select
      style={{ width: 300 }}
      placeholder="custom dropdown render"
      dropdownRender={(menu) => (
        <>
          {menu}
          <Divider style={{ margin: '8px 0' }}
          <Space style={{ padding: '0 8px 4px'
            <Input
              placeholder="Please enter item"
              ref={inputRef}
              value={name}
              onChange={onNameChange}
            />
            <Button type="text" icon={<PlusOut
              Add item
            </Button>
          </Space>
        </>
      )}
      options={items.map((item) => ({ label: i
    />
  );
};

export default App;
```

Hide Already Selected

Hide already selected options in the dropdown.

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

const OPTIONS = ['Apples', 'Nails', 'Bananas',

const App: React.FC = () => {
  const [selectedItems, setSelectedItems] = us

  const filteredOptions = OPTIONS.filter((o) =

  return (
    <Select
      mode="multiple"
      placeholder="Inserted are removed"
      value={selectedItems}
      onChange={setSelectedItems}
      style={{ width: '100%' }}
      options={filteredOptions.map((item) => (
        value: item,
        label: item,
      }))}
    />
  );
};

export default App;
```

Bordered-less

Bordered-less style component.

```
import { Select, Space } from 'antd';
import React from 'react';

const App: React.FC = () => (
  <Space wrap>
    <Select
      defaultValue="lucy"
      style={{ width: 120 }}
      bordered={false}
      options={[
        { value: 'jack', label: 'Jack' },
        { value: 'lucy', label: 'Lucy' },
        { value: 'Yiminghe', label: 'yiminghe'
      ]}
    />
    <Select
      defaultValue="lucy"
      style={{ width: 120 }}
      disabled
      bordered={false}
      options={[{ value: 'lucy', label: 'Lucy'
    />
  </Space>
);

export default App;
```

Custom Tag Render

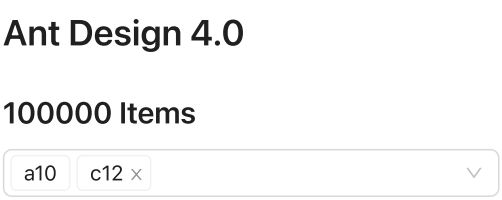
Allows for custom rendering of tags.

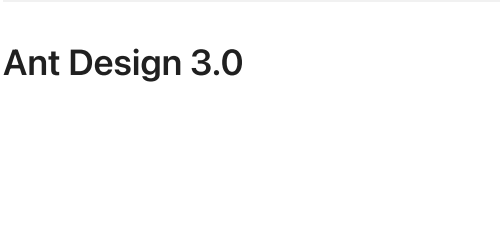
```
import React from 'react';
import { Select, Tag } from 'antd';
import type { CustomTagProps } from 'rc-select

const options = [{ value: 'gold' }, { value: '

const tagRender = (props: CustomTagProps) => {
  const { label, value, closable, onClose } =
  const onPreventMouseDown = (event: React.Mou
    event.preventDefault();
    event.stopPropagation();
  };
  return (
    <Tag
      color={value}
      onMouseDown={onPreventMouseDown}
      closable={closable}
      onClose={onClose}
      style={{ marginRight: 3 }}
    >
      {label}
    </Tag>
  );
};

const App: React.FC = () => (
  <Select
    mode="multiple"
    showArrow
    tagRender={tagRender}
    defaultValue={['gold', 'cyan']}
    style={{ width: '100%' }}
    options={options}
  />
);

export default App;
```

Responsive maxTagCount

Auto collapse to tag with responsive case. Not recommend use in large form case since responsive calculation has a perf cost.

```
import React, { useState } from 'react';
import type { SelectProps } from 'antd';
import { Select, Space } from 'antd';

interface ItemProps {
  label: string;
  value: string;
}

const options: ItemProps[] = [];

for (let i = 10; i < 36; i++) {
  const value = i.toString(36) + i;
  options.push({
    label: `Long Label: ${value}`,
    value,
  });
}

const App: React.FC = () => {
  const [value, setValue] = useState(['a10', '

  const selectProps: SelectProps = {
    mode: 'multiple',
    style: { width: '100%' },
    value,
    options,
    onChange: (newValue: string[]) => {
      setValue(newValue);
    },
    placeholder: 'Select Item...',
    maxTagCount: 'responsive',
  };

  return (
    <Space direction="vertical" style={{ width
      <Select {...selectProps} />
      <Select {...selectProps} disabled />
    </Space>
  );
};

export default App;
```

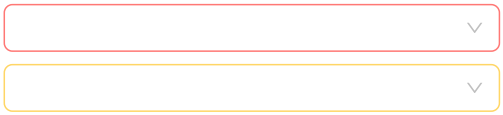




Big Data

Select use virtual scroll which get better performance than 3.0.

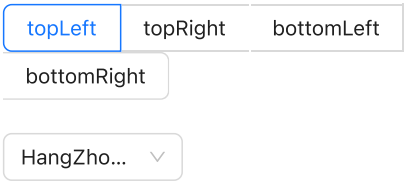
```
import React from 'react';
import type { SelectProps } from 'antd';
import { Divider, Select, Typography } from 'a

const { Title } = Typography;

const options: SelectProps['options'] = [];

for (let i = 0; i < 100000; i++) {
  const value = `${i.toString(36)}${i}`;
  options.push({
    label: value,
    value,
    disabled: i === 10,
  });
}

const handleChange = (value: string[]) => {
  console.log(`selected ${value}`);
};

const App: React.FC = () => (
  <>
    <Title level={3}>Ant Design 4.0</Title>
    <Title level={4}>{options.length} Items</T
    <Select
      mode="multiple"
      style={{ width: '100%' }}
      placeholder="Please select"
      defaultValue={['a10', 'c12']}
      onChange={handleChange}
      options={options}
    />

    <Divider />

    <Title level={3}>Ant Design 3.0</Title>
```



Status

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

```
import React from 'react';
import { Select, Space } from 'antd';

const App: React.FC = () => (
  <Space direction="vertical" style={{ width:
    <Select status="error" style={{ width: '10
    <Select status="warning" style={{ width: '
  </Space>
);

export default App;
```



Placement

You can manually specify the position of the popup via `placement`.

```
import React, { useState } from 'react';
import type { RadioChangeEvent } from 'antd';
import { Radio, Select } from 'antd';
import type { SelectCommonPlacement } from 'ar

const App: React.FC = () => {
  const [placement, SetPlacement] = useState<S

  const placementChange = (e: RadioChangeEvent
    SetPlacement(e.target.value);
  };

  return (
    <>
      <Radio.Group value={placement} onChange=
        <Radio.Button value="topLeft">topLeft<
        <Radio.Button value="topRight">topRigh
        <Radio.Button value="bottomLeft">bottc
        <Radio.Button value="bottomRight">bott
      </Radio.Group>
      <br />
      <br />
      <Select
        defaultValue="HangZhou"
        style={{ width: 120 }}
        dropdownMatchSelectWidth={false}
        placement={placement}
        options={[
          {
            value: 'HangZhou',
            label: 'HangZhou #310000',
          },
          {
            value: 'NingBo',
            label: 'NingBo #315000',
          },
          {
            value: 'WenZhou',
            label: 'WenZhou #325000',
          },
        ]}
      />
    </>
  );
};

export default App;
```

API

```
<Select>
  <Option value="lucy">lucy</Option>
```

```
</Select>
```

Select props

Property	Description	Type	Default
allowClear	Show clear button	boolean	false
autoClearSearchValue	Whether the current search will be cleared on selecting an item. Only applies when `mode` is set to `multiple` or `tags`	boolean	true
autoFocus	Get focus by default	boolean	false
bordered	Whether has border style	boolean	true
clearIcon	The custom clear icon	ReactNode	-
defaultActiveFirstOption	Whether active first option by default	boolean	true
defaultOpen	Initial open state of dropdown	boolean	-
defaultValue	Initial selected option	string | string[] | number | number[] | LabeledValue | LabeledValue[]	-
disabled	Whether disabled select	boolean	false
popupClassName	The className of dropdown menu	string	-
dropdownMatchSelectWidth	Determine whether the dropdown menu and the select input are the same width. Default set `min-width` same as input. Will ignore when value less than select width. `false` will disable virtual scroll	boolean | number	true
dropdownRender	Customize dropdown content	(originNode: ReactNode) => ReactNode	-
dropdownStyle	The style of dropdown menu	CSSProperties	-
fieldNames	Customize node label, value, options field name	object	{ label: `label`, value:

Property	Description	Type	Default
			`value`, options: `options`
filterOption	If true, filter options by input, if function, filter options against it. The function will receive two arguments, `inputValue` and `option`, if the function returns `true`, the option will be included in the filtered set; Otherwise, it will be excluded	boolean | function(inputValue, option)	true
filterSort	Sort function for search options sorting, see Array.sort's compareFunction	(optionA: Option, optionB: Option) => number	-
getPopupContainer	Parent Node which the selector should be rendered to. Default to `body`. When position issues happen, try to modify it into scrollable content and position it relative. Example	function(triggerNode)	() => document
labelInValue	Whether to embed label in value, turn the format of value from `string` to { value: string, label: ReactNode }	boolean	false
listHeight	Config popup height	number	256
loading	Indicate loading state	boolean	false
maxTagCount	Max tag count to show. `responsive` will cost render performance	number | `responsive`	-
maxTagPlaceholder	Placeholder for not showing tags	ReactNode | function(omittedValues)	-
maxTagTextLength	Max tag text length to show	number	-
menuItemSelectedIcon	The custom menuItemSelected icon with multiple options	ReactNode	-
mode	Set mode of Select	`multiple` | `tags`	-

Property	Description	Type	Default
notFoundContent	Specify content to show when no result matches	ReactNode	`Not Found`
open	Controlled open state of dropdown	boolean	-
optionFilterProp	Which prop value of option will be used for filter if filterOption is true. If `options` is set, it should be set to `label`	string	`value`
optionLabelProp	Which prop value of option will render as content of select. Example	string	`children`
options	Select options. Will get better perf than jsx definition	{ label, value }[]	-
placeholder	Placeholder of select	ReactNode	-
placement	The position where the selection box pops up	`bottomLeft` `bottomRight` `topLeft` `topRight`	bottomLe
removeIcon	The custom remove icon	ReactNode	-
searchValue	The current input "search" text	string	-
showArrow	Whether to show the drop-down arrow	boolean	`true`
showSearch	Whether select is searchable	boolean	single: false, multiple: true
size	Size of Select input	`large` | `middle` | `small`	`middle`
status	Set validation status	'error' | 'warning'	-
suffixIcon	The custom suffix icon	ReactNode	-
tagRender	Customize tag render, only applies when `mode` is set to `multiple` or `tags`	(props) => ReactNode	-
tokenSeparators	Separator used to tokenize, only	string[]	-

Property	Description	Type	Default
	applies when `mode="tags"`		
value	Current selected option (considered as a immutable array)	string | string[] | number | number[] | LabeledValue | LabeledValue[]	-
virtual	Disable virtual scroll when set to false	boolean	true
onBlur	Called when blur	function	-
onChange	Called when select an option or input value change	function(value, option:Option | Array<Option>)	-
onClear	Called when clear	function	-
onDeselect	Called when an option is deselected, param is the selected option's value. Only called for `multiple` or `tags`, effective in multiple or tags mode only	function(value: string | number | LabeledValue)	-
onDropdownVisibleChange	Called when dropdown open	function(open)	-
onFocus	Called when focus	function	-
onInputKeyDown	Called when key pressed	function	-
onMouseEnter	Called when mouse enter	function	-
onMouseLeave	Called when mouse leave	function	-
onPopupScroll	Called when dropdown scrolls	function	-
onSearch	Callback function that is fired when input changed	function(value: string)	-
onSelect	Called when an option is selected, the params are option's value (or key) and option instance	function(value: string | number | LabeledValue, option: Option)	-

> Note, if you find that the drop-down menu scrolls with the page, or you need to trigger Select in other popup layers, please try to use `getPopupContainer={triggerNode => triggerNode.parentElement}` to fix the drop-down popup rendering node in the parent element of the trigger .

Select Methods

Name	Description	Version
blur()	Remove focus	
focus()	Get focus	

Option props

Property	Description	Type	Default	Version
className	The additional class to option	string	-	
disabled	Disable this option	boolean	false	
title	`title` attribute of Select Option	string	-	
value	Default to filter with this property	string | number	-	

OptGroup props

Property	Description	Type	Default	Version
key	Group key	string	-	
label	Group label	string | React.Element	-	

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`.	`string`	`#ffffff`
colorBgContainerDisabled	Control the background color of container in disabled state.	`string`	`rgba(0, 0, 0, 0.04)`
colorBgElevated	Container background color of the popup layer, in dark mode the color value of this token will be a little brighter than `colorBgContainer`. E.g: modal, pop-up, menu, etc.	`string`	`#ffffff`
colorBorder	Default border color, used to separate different elements, such as: form separator, card separator, etc.	`string`	`#d9d9d9`
colorErrorHover	The hover state of the error color.	`string`	`#ff7875`

Token Name	Description	Type	Default Value
colorErrorOutline	Control the outline color of input component in error state.	`string`	rgba(255, 38, 5, 0.06)
colorFillSecondary	The second level of fill color can outline the shape of the element more clearly, such as Rate, Skeleton, etc. It can also be used as the Hover state of the third level of fill color, such as Table, etc.	`string`	rgba(0, 0, 0, 0.06)
colorIcon	Weak action. Such as `allowClear` or Alert close button	`string`	rgba(0, 0, 0, 0.45)
colorIconHover	Weak action hover color. Such as `allowClear` or Alert close button	`string`	rgba(0, 0, 0, 0.88)
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`	#1677ff
colorPrimaryHover	Hover state under the main color gradient.	`string`	#4096ff
colorSplit	Used as the color of separator, this color is the same as colorBorderSecondary but with transparency.	`string`	rgba(5, 5, 5, 0.06)
colorText	Default text color which comply with W3C standards, and this color is also the darkest neutral color.	`string`	rgba(0, 0, 0, 0.88)
colorTextDescription	Control the font color of text description.	`string`	rgba(0, 0, 0, 0.45)
colorTextDisabled	Control the color of text in disabled state.	`string`	rgba(0, 0, 0, 0.25)
colorTextPlaceholder	Control the color of placeholder text.	`string`	rgba(0, 0, 0, 0.25)
colorTextQuaternary	The fourth level of text color is the lightest text color, such as form input prompt text, disabled color text, etc.	`string`	rgba(0, 0, 0, 0.25)
colorTextTertiary	The third level of text color is generally used for descriptive text, such as form supplementary explanation text, list descriptive text, etc.	`string`	rgba(0, 0, 0, 0.45)
colorWarningHover	The hover state of the warning color.	`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

Token Name	Description	Type	Default Value
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
borderRadiusXS	XS size border radius, used in some small border radius components, such as Segmented, Arrow and other components with small border radius.	`number`	2
boxShadowSecondary	Control the secondary box shadow style of an element.	`string`	0 6px 16px 0 rgba(0, 0, 0, 0.08), 0 3px 6px -4px rgba(0, 0, 0, 0.12), 0 9px 28px 8px rgba(0, 0, 0, 0.05)
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
controlHeightXS	XS component height	`number`	16
controlItemBgActive	Control the background color of control component item when active.	`string`	`#e6f4ff`
controlItemBgHover	Control the background color of control component item when hovering.	`string`	`rgba(0, 0, 0, 0.04)`
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

Token Name	Description	Type	Default Value
fontSizeIcon	Control the font size of operation icon in Select, Cascader, etc. Normally same as fontSizeSM.	`number`	12
fontSizeLG	Large font size	`number`	16
fontSizeSM	Small font size	`number`	12
fontWeightStrong	Control the font weight of heading components (such as h1, h2, h3) or selected item.	`number`	600
lineHeight	Line height of text.	`number`	1.5714285714285714
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
motionEaseInOut	Preset motion curve.	`string`	cubic-bezier(0.645, 0.045, 0.355, 1)
motionEaseInOutCirc	Preset motion curve.	`string`	cubic-bezier(0.78, 0.14, 0.15, 0.86)
motionEaseInQuint	Preset motion curve.	`string`	cubic-bezier(0.755, 0.05, 0.855, 0.06)
motionEaseOutCirc	Preset motion curve.	`string`	cubic-bezier(0.08, 0.82, 0.17, 1)
motionEaseOutQuint	Preset motion curve.	`string`	cubic-bezier(0.23, 1, 0.32, 1)
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
zIndexPopupBase	Base zIndex of component like FloatButton, Affix which can be cover by large popup	`number`	1000

FAQ

Why sometime search will show 2 same option when in `tags` mode?

It's caused by option with different `label` and `value`. You can use `optionFilterProp="label"` to change filter logic instead.

When I click elements in dropdownRender, the select dropdown will not be closed?

You can control it by `open` prop: codesandbox.

I don't want dropdown close when click inside `dropdownRender`?

Select will close when it lose focus. You can prevent event to handle this:

```
<Select
  dropdownRender={() => (
    <div
      onMouseDown={(e) => {
        e.preventDefault();
        e.stopPropagation();
      }}
    >
      Some Content
    </div>
  )}
/>
```

Why sometime customize Option cause scroll break?

Virtual scroll internal set item height as `24px`. You need to adjust `listItemHeight` when your option height is less and `listHeight` config list container height:

```
<Select listItemHeight={10} listHeight={250} />
```

Note: `listItemHeight` and `listHeight` are internal props. Please only modify when necessary.

Why a11y test report missing `aria-` props?

Select only create a11y auxiliary node when operating on. Please open Select and retry. For `aria-label` & `aria-labelledby` miss warning, please add related prop to Select with your own requirement.

Default virtual scrolling will create a mock element to simulate an accessible binding. If a screen reader needs to fully access the entire list, you can set `virtual={false}` to disable virtual scrolling and the accessibility option will be bound to the actual element.