When To Use

`TreeSelect` is similar to `Select`, but the values are provided in a tree like structure. Any data whose entries are defined in a hierarchical manner is fit to use this control. Examples of such case may include a corporate hierarchy, a directory structure, and so on.

Examples

Basic

```
import React, { useState } from 'react';
import { TreeSelect } from 'antd';
import type { TreeSelectProps } from 'antd';

const treeData = [
  {
    value: 'parent 1',
    title: 'parent 1',
    children: [
      {
        value: 'parent 1-0',
        title: 'parent 1-0',
        children: [
          {
            value: 'leaf1',
            title: 'leaf1',
          },
          {
            value: 'leaf2',
            title: 'leaf2',
          },
          {
            value: 'leaf3',
            title: 'leaf3',
          },
          {
            value: 'leaf4',
            title: 'leaf4',
          },
          {
            value: 'leaf5',
            title: 'leaf5',
          },
          {
            value: 'leaf6',
            title: 'leaf6',
```

```
          },
        ],
      },
      {
        value: 'parent 1-1',
        title: 'parent 1-1',
        children: [
          {
            value: 'leaf11',
            title: <b style={{ color: '#08c' }}>leaf11</b>,
          },
        ],
      },
    ],
  },
];
const App: React.FC = () => {
  const [value, setValue] = useState<string>();

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

  const onPopupScroll: TreeSelectProps['onPopupScroll'] = (e) => {
    console.log('onPopupScroll', e);
  };

  return (
    <TreeSelect
      showSearch
      style={{ width: '100%' }}
      value={value}
      dropdownStyle={{ maxHeight: 400, overflow: 'auto' }}
      placeholder="Please select"
      allowClear
      treeDefaultExpandAll
      onChange={onChange}
      treeData={treeData}
      onPopupScroll={onPopupScroll}
    />
  );
};

export default App;
```

Multiple Selection

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

const treeData = [
  {
    value: 'parent 1',
    title: 'parent 1',
    children: [
      {
        value: 'parent 1-0',
        title: 'parent 1-0',
        children: [
          {
            value: 'leaf1',
            title: 'my leaf',
          },
          {
            value: 'leaf2',
            title: 'your leaf',
          },
        ],
      },
      {
        value: 'parent 1-1',
        title: 'parent 1-1',
        children: [
          {
            value: 'sss',
            title: <b style={{ color: '#08c' }}>sss</b>,
          },
        ],
      },
    ],
  },
];
const App: React.FC = () => {
  const [value, setValue] = useState<string>();

  const onChange = (newValue: string) => {
    console.log(newValue);
    setValue(newValue);
  };

  return (
    <TreeSelect
      showSearch
```

```
      style={{ width: '100%' }}
      value={value}
      dropdownStyle={{ maxHeight: 400, overflow: 'auto' }}
      placeholder="Please select"
      allowClear
      multiple
      treeDefaultExpandAll
      onChange={onChange}
      treeData={treeData}
    />
  );
};

export default App;
```

Generate from tree data

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

const treeData = [
  {
    title: 'Node1',
    value: '0-0',
    children: [
      {
        title: 'Child Node1',
        value: '0-0-1',
      },
      {
        title: 'Child Node2',
        value: '0-0-2',
      },
    ],
  },
  {
    title: 'Node2',
    value: '0-1',
  },
];

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

  const onChange = (newValue: string) => {
    console.log(newValue);
```

```
    setValue(newValue);
  };

  return (
    <TreeSelect
      style={{ width: '100%' }}
      value={value}
      dropdownStyle={{ maxHeight: 400, overflow: 'auto' }}
      treeData={treeData}
      placeholder="Please select"
      treeDefaultExpandAll
      onChange={onChange}
    />
  );
};

export default App;
```

Checkable

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

const { SHOW_PARENT } = TreeSelect;

const treeData = [
  {
    title: 'Node1',
    value: '0-0',
    key: '0-0',
    children: [
      {
        title: 'Child Node1',
        value: '0-0-0',
        key: '0-0-0',
      },
    ],
  },
  {
    title: 'Node2',
    value: '0-1',
    key: '0-1',
    children: [
      {
        title: 'Child Node3',
        value: '0-1-0',
```

```
        key: '0-1-0',
      },
      {
        title: 'Child Node4',
        value: '0-1-1',
        key: '0-1-1',
      },
      {
        title: 'Child Node5',
        value: '0-1-2',
        key: '0-1-2',
      },
    ],
  },
];

const App: React.FC = () => {
  const [value, setValue] = useState(['0-0-0']);

  const onChange = (newValue: string[]) => {
    console.log('onChange ', newValue);
    setValue(newValue);
  };

  const tProps = {
    treeData,
    value,
    onChange,
    treeCheckable: true,
    showCheckedStrategy: SHOW_PARENT,
    placeholder: 'Please select',
    style: {
      width: '100%',
    },
  };

  return <TreeSelect {...tProps} />;
};

export default App;
```

Asynchronous loading

```
import React, { useState } from 'react';
import type { GetProp, TreeSelectProps } from 'antd';
import { TreeSelect } from 'antd';
```

```
type DefaultOptionType = GetProp<TreeSelectProps, 'treeData'>[number];

const App: React.FC = () => {
  const [value, setValue] = useState<string>();
  const [treeData, setTreeData] = useState<Omit<DefaultOptionType, 'label'>
[]>([
    { id: 1, pId: 0, value: '1', title: 'Expand to load' },
    { id: 2, pId: 0, value: '2', title: 'Expand to load' },
    { id: 3, pId: 0, value: '3', title: 'Tree Node', isLeaf: true },
  ]);

  const genTreeNode = (parentId: number, isLeaf = false) => {
    const random = Math.random().toString(36).substring(2, 6);
    return {
      id: random,
      pId: parentId,
      value: random,
      title: isLeaf ? 'Tree Node' : 'Expand to load',
      isLeaf,
    };
  };

  const onLoadData: TreeSelectProps['loadData'] = ({ id }) =>
    new Promise((resolve) => {
      setTimeout(() => {
        setTreeData(
          treeData.concat([genTreeNode(id, false), genTreeNode(id, true),
genTreeNode(id, true)]),
        );
        resolve(undefined);
      }, 300);
    });

  const onChange = (newValue: string) => {
    console.log(newValue);
    setValue(newValue);
  };

  return (
    <TreeSelect
      treeDataSimpleMode
      style={{ width: '100%' }}
      value={value}
      dropdownStyle={{ maxHeight: 400, overflow: 'auto' }}
      placeholder="Please select"
```

```
      onChange={onChange}
      loadData={onLoadData}
      treeData={treeData}
    />
  );
};

export default App;
```

Show Tree Line

```
import React, { useState } from 'react';
import { CarryOutOutlined } from '@ant-design/icons';
import { Space, Switch, TreeSelect } from 'antd';

const treeData = [
  {
    value: 'parent 1',
    title: 'parent 1',
    icon: <CarryOutOutlined />,
    children: [
      {
        value: 'parent 1-0',
        title: 'parent 1-0',
        icon: <CarryOutOutlined />,
        children: [
          {
            value: 'leaf1',
            title: 'leaf1',
            icon: <CarryOutOutlined />,
          },
          {
            value: 'leaf2',
            title: 'leaf2',
            icon: <CarryOutOutlined />,
          },
        ],
      },
      {
        value: 'parent 1-1',
        title: 'parent 1-1',
        icon: <CarryOutOutlined />,
        children: [
          {
            value: 'sss',
            title: 'sss',
```

```
            icon: <CarryOutOutlined />,
          },
        ],
      },
    ],
  },
];

const App: React.FC = () => {
  const [treeLine, setTreeLine] = useState(true);
  const [showLeafIcon, setShowLeafIcon] = useState(false);
  const [showIcon, setShowIcon] = useState<boolean>(false);

  return (
    <Space direction="vertical">
      <Switch
        checkedChildren="showIcon"
        unCheckedChildren="showIcon"
        checked={showIcon}
        onChange={() => setShowIcon(!showIcon)}
      />
      <Switch
        checkedChildren="treeLine"
        unCheckedChildren="treeLine"
        checked={treeLine}
        onChange={() => setTreeLine(!treeLine)}
      />
      <Switch
        disabled={!treeLine}
        checkedChildren="showLeafIcon"
        unCheckedChildren="showLeafIcon"
        checked={showLeafIcon}
        onChange={() => setShowLeafIcon(!showLeafIcon)}
      />
      <TreeSelect
        treeLine={treeLine && { showLeafIcon }}
        style={{ width: 300 }}
        treeData={treeData}
        treeIcon={showIcon}
      />
    </Space>
  );
};

export default App;
```

Placement

```
import React, { useState } from 'react';
import type { GetProp, RadioChangeEvent, TreeSelectProps } from 'antd';
import { Radio, TreeSelect } from 'antd';

type SelectCommonPlacement = GetProp<TreeSelectProps, 'placement'>;

const treeData = [
  {
    value: 'parent 1',
    title: 'parent 1',
    children: [
      {
        value: 'parent 1-0',
        title: 'parent 1-0',
        children: [
          {
            value: 'leaf1',
            title: 'leaf1',
          },
          {
            value: 'leaf2',
            title: 'leaf2',
          },
        ],
      },
      {
        value: 'parent 1-1',
        title: 'parent 1-1',
        children: [
          {
            value: 'leaf3',
            title: <b style={{ color: '#08c' }}>leaf3</b>,
          },
        ],
      },
    ],
  },
];
const App: React.FC = () => {
  const [placement, SetPlacement] = useState<SelectCommonPlacement>
('topLeft');

  const placementChange = (e: RadioChangeEvent) => {
    SetPlacement(e.target.value);
```

```
  };

  return (
    <>
      <Radio.Group value={placement} onChange={placementChange}>
        <Radio.Button value="topLeft">topLeft</Radio.Button>
        <Radio.Button value="topRight">topRight</Radio.Button>
        <Radio.Button value="bottomLeft">bottomLeft</Radio.Button>
        <Radio.Button value="bottomRight">bottomRight</Radio.Button>
      </Radio.Group>
      <br />
      <br />

      <TreeSelect
        showSearch
        dropdownStyle={{ maxHeight: 400, overflow: 'auto', minWidth: 300 }}
        placeholder="Please select"
        popupMatchSelectWidth={false}
        placement={placement}
        allowClear
        treeDefaultExpandAll
        treeData={treeData}
      />
    </>
  );
};

export default App;
```

Variants

v5.13.0

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

const style: React.CSSProperties = {
  width: '100%',
  maxWidth: '100%',
};

const App: React.FC = () => {
  return (
    <Flex vertical gap="middle">
      <TreeSelect style={style} placeholder="Please select"
variant="borderless" />
```

```
      <TreeSelect style={style} placeholder="Please select"
variant="filled" />
      <TreeSelect style={style} placeholder="Please select"
variant="outlined" />
      <TreeSelect style={style} placeholder="Please select"
variant="underlined" />
    </Flex>
  );
};

export default App;
```

Status

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

const App: React.FC = () => (
  <Space direction="vertical" style={{ width: '100%' }}>
    <TreeSelect status="error" style={{ width: '100%' }}
placeholder="Error" />
    <TreeSelect
      status="warning"
      style={{ width: '100%' }}
      multiple
      placeholder="Warning multiple"
    />
  </Space>
);

export default App;
```

Max Count

v5.23.0

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

const MAX_COUNT = 3;

const treeData = [
  {
    title: 'Parent 1',
    value: 'parent1',
```

```
    children: [
      {
        title: 'Child 1-1',
        value: 'child1-1',
      },
      {
        title: 'Child 1-2',
        value: 'child1-2',
      },
    ],
  },
  {
    title: 'Parent 2',
    value: 'parent2',
    children: [
      {
        title: 'Child 2-1',
        value: 'child2-1',
      },
      {
        title: 'Child 2-2',
        value: 'child2-2',
      },
    ],
  },
];

const App: React.FC = () => {
  const [value, setValue] = React.useState<string[]>(['child1-1']);

  const onChange = (newValue: string[]) => {
    setValue(newValue);
  };

  const suffix = (
    <>
      <span>
        {value.length} / {MAX_COUNT}
      </span>
      <DownOutlined />
    </>
  );

  return (
    <TreeSelect
      treeData={treeData}
```

```
      value={value}
      onChange={onChange}
      multiple
      maxCount={MAX_COUNT}
      style={{ width: '100%' }}
      suffixIcon={suffix}
      treeCheckable
      placeholder="Please select"
      showCheckedStrategy={TreeSelect.SHOW_CHILD}
    />
  );
};

export default App;
```

Prefix and Suffix

v5.22.0

```
import React, { useState } from 'react';
import { SmileOutlined } from '@ant-design/icons';
import { TreeSelect } from 'antd';

const icon = <SmileOutlined />;
const treeData = [
  {
    value: 'parent 1',
    title: 'parent 1',
    children: [
      {
        value: 'parent 1-0',
        title: 'parent 1-0',
        children: [
          {
            value: 'leaf1',
            title: 'my leaf',
          },
          {
            value: 'leaf2',
            title: 'your leaf',
          },
        ],
      },
      {
        value: 'parent 1-1',
        title: 'parent 1-1',
```

```
        children: [
          {
            value: 'sss',
            title: <b style={{ color: '#08c' }}>sss</b>,
          },
        ],
      },
    ],
  },
];

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

  const onChange = (newValue: string) => {
    console.log(newValue);
    setValue(newValue);
  };

  return (
    <>
      <TreeSelect
        showSearch
        suffixIcon={icon}
        style={{ width: '100%' }}
        value={value}
        dropdownStyle={{ maxHeight: 400, overflow: 'auto' }}
        placeholder="Please select"
        allowClear
        treeDefaultExpandAll
        onChange={onChange}
        treeData={treeData}
      />
      <br />
      <br />
      <TreeSelect
        showSearch
        prefix="Prefix"
        style={{ width: '100%' }}
        value={value}
        dropdownStyle={{ maxHeight: 400, overflow: 'auto' }}
        placeholder="Please select"
        allowClear
        treeDefaultExpandAll
        onChange={onChange}
        treeData={treeData}
```

```
      />
    </>
  );
};

export default App;
```

_InternalPanelDoNotUseOrYouWillBeFired

Debug

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

const { _InternalPanelDoNotUseOrYouWillBeFired: InternalTreeSelect } =
TreeSelect;

const treeData = [
  {
    title: 'Node1',
    value: '0-0',
    children: [
      {
        title: 'Child Node1',
        value: '0-0-1',
      },
      {
        title: 'Child Node2',
        value: '0-0-2',
      },
    ],
  },
  {
    title: 'Node2',
    value: '0-1',
  },
];

const App: React.FC = () => (
  <InternalTreeSelect defaultValue="lucy" style={{ width: '100%' }}
treeData={treeData} />
);

export default App;
```

Component Token

Debug

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

const treeData = [
  {
    value: 'parent 1',
    title: 'parent 1',
    children: [
      {
        value: 'parent 1-0',
        title: 'parent 1-0',
        children: [
          {
            value: 'leaf1',
            title: 'leaf1',
          },
          {
            value: 'leaf2',
            title: 'leaf2',
          },
        ],
      },
      {
        value: 'parent 1-1',
        title: 'parent 1-1',
        children: [
          {
            value: 'leaf3',
            title: <b style={{ color: '#08c' }}>leaf3</b>,
          },
        ],
      },
    ],
  },
];
const App: React.FC = () => {
  const [value, setValue] = useState<string>();

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

  return (
    <ConfigProvider
```

```
      theme={{
        components: {
          TreeSelect: {
            nodeHoverBg: '#fff2f0',
            nodeSelectedBg: '#ffa39e',
          },
        },
      }}
    >
      <TreeSelect
        showSearch
        style={{ width: '100%' }}
        value={value}
        dropdownStyle={{ maxHeight: 400, overflow: 'auto' }}
        placeholder="Please select"
        allowClear
        treeDefaultExpandAll
        onChange={onChange}
        treeData={treeData}
      />
    </ConfigProvider>
  );
};

export default App;
```

API

Common props ref：Common props

Tree props

Property	Description	Type	
allowClear	Customize clear icon	boolean | { clearIcon?: ReactNode }	fal
autoClearSearchValue	If auto clear search input value when multiple select is selected/deselected	boolean	tru
defaultOpen	Initial open state of dropdown	boolean	-
defaultValue	To set the initial selected treeNode(s)	string | string[]	-

disabled	Disabled or not	boolean	fal
popupClassName	The className of dropdown menu	string	-
popupMatchSelectWidth	Determine whether the popup 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	tru
dropdownRender	Customize dropdown content	(originNode: ReactNode, props) => ReactNode	-
dropdownStyle	To set the style of the dropdown menu	CSSProperties	-
fieldNames	Customize node label, value, children field name	object	{ l va ch
filterTreeNode	Whether to filter treeNodes by input value. The value of `treeNodeFilterProp` is used for filtering by default	boolean | function(inputValue: string, treeNode: TreeNode) (should return boolean)	fu
getPopupContainer	To set the container of the dropdown menu. The default is to create a `div` element in `body`, you can reset it to the scrolling area and make a relative reposition. example	function(triggerNode)	()
labelInValue	Whether to embed label in value, turn the format of value from `string` to {value: string, label: ReactNode, halfChecked: string[]}	boolean	fal
listHeight	Config popup height	number	25
loadData	Load data asynchronously. Will not load when filtering. Check FAQ for more info	function(node)	-

maxTagCount	Max tag count to show. `responsive` will cost render performance	number | `responsive`	-
maxCount	The maximum number of items that can be selected. Only takes effect when `multiple=true`. If (`showCheckedStrategy = 'SHOW_ALL'` and `treeCheckStrictly` is disabled) or `showCheckedStrategy = 'SHOW_PARENT'` is used, `maxCount` will not take effect.	number	-
maxTagPlaceholder	Placeholder for not showing tags	ReactNode | function(omittedValues)	-
maxTagTextLength	Max tag text length to show	number	-
multiple	Support multiple or not, will be `true` when enable `treeCheckable`	boolean	fa
notFoundContent	Specify content to show when no result matches	ReactNode	No
open	Controlled open state of dropdown	boolean	-
placeholder	Placeholder of the select input	string	-
placement	The position where the selection box pops up	`bottomLeft` `bottomRight` `topLeft` `topRight`	bo
prefix	The custom prefix	ReactNode	-
searchValue	Work with `onSearch` to make search value controlled	string	-
showCheckedStrategy	The way show selected item in box when `treeCheckable` set. **Default:** just show child nodes.	`TreeSelect.SHOW_ALL` | `TreeSelect.SHOW_PARENT` | `TreeSelect.SHOW_CHILD`	Tr

	`TreeSelect.SHOW_ALL`: show all checked treeNodes (include parent treeNode). **`TreeSelect.SHOW_PARENT`:** show checked treeNodes (just show parent treeNode)		
showSearch	Support search or not	boolean	si tr
size	To set the size of the select input	`large` | `middle` | `small`	-
status	Set validation status	'error' | 'warning'	-
suffixIcon	The custom suffix icon	ReactNode	<D
switcherIcon	Customize collapse/expand icon of tree node	ReactNode | ((props: AntTreeNodeProps) => ReactNode)	-
tagRender	Customize tag render when `multiple`	(props) => ReactNode	-
treeCheckable	Whether to show checkbox on the treeNodes	boolean	fa
treeCheckStrictly	Whether to check nodes precisely (in the `checkable` mode), means parent and child nodes are not associated, and it will make `labelInValue` be true	boolean	fa
treeData	Data of the treeNodes, manual construction work is no longer needed if this property has been set(ensure the Uniqueness of each value)	array<{ value, title, children, [disabled, disableCheckbox, selectable, checkable] }>	[]
treeDataSimpleMode	Enable simple mode of treeData. Changes the `treeData` schema to: [{id:1, pId:0, value:'1', title:"test1",...},...] where pId is parent node's id). It is possible to replace the default `id` and `pId` keys by	boolean | object<{ id: string, pId: string, rootPId: string }>	fa

	providing object to `treeDataSimpleMode`		
treeTitleRender	Customize tree node title render	(nodeData) => ReactNode	-
treeDefaultExpandAll	Whether to expand all treeNodes by default	boolean	fal
treeDefaultExpandedKeys	Default expanded treeNodes	string[]	-
treeExpandAction	Tree title open logic when click, optional: false | `click` | `doubleClick`	string | boolean	fal
treeExpandedKeys	Set expanded keys	string[]	-
treeIcon	Shows the icon before a TreeNode's title. There is no default style; you must set a custom style for it if set to `true`	boolean	fal
treeLoadedKeys	(Controlled) Set loaded tree nodes, work with `loadData` only	string[]	[]
treeLine	Show the line. Ref Tree - showLine	boolean | object	fal
treeNodeFilterProp	Will be used for filtering if `filterTreeNode` returns true	string	va
treeNodeLabelProp	Will render as content of select	string	`ti`
value	To set the current selected treeNode(s)	string | string[]	-
variant	Variants of selector	`outlined` | `borderless` | `filled` | `underlined`	ou
virtual	Disable virtual scroll when set to false	boolean	tru
onChange	A callback function, can be executed when selected	function(value, label, extra)	-

	treeNodes or input value change		
onDropdownVisibleChange	Called when dropdown open	function(open)	-
onSearch	A callback function, can be executed when the search input changes	function(value: string)	-
onSelect	A callback function, can be executed when you select a treeNode	function(value, node, extra)	-
onTreeExpand	A callback function, can be executed when treeNode expanded	function(expandedKeys)	-
onPopupScroll	Called when dropdown scrolls	(event: UIEvent) => void	-

Tree Methods

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

TreeNode props

> *We recommend you to use `treeData` rather than `TreeNode`, to avoid the trouble of manual construction.*

Property	Description	Type	Default	Version
checkable	When Tree is checkable, set TreeNode display Checkbox or not	boolean	-	
disableCheckbox	Disables the checkbox of the treeNode	boolean	false	
disabled	Disabled or not	boolean	false	
isLeaf	Leaf node or not	boolean	false	
key	Required property (unless using `treeDataSimpleMode`), should be unique in the tree	string	-	
selectable	Whether can be selected	boolean	true	
title	Content showed on the treeNodes	ReactNode	---	

value	Will be treated as `treeNodeFilterProp` by default, should be unique in the tree	string	-	

Design Token

FAQ

How to get parent node in onChange?

We don't provide this since performance consideration. You can get by this way: https://codesandbox.io/s/get-parent-node-in-onchange-eb1608

Why sometime customize Option cause scroll break?

You can ref Select FAQ.

Why `loadData` not trigger when searching?

In earlier version, `loadData` will be triggered when searching. But we got feedback that it will block network when inputting. So we change it to not trigger `loadData` when searching. But you can still handle async logic by `filterTreeNode`:

```
<TreeSelect
  filterTreeNode={(input, treeNode) => {
    const match = YOUR_LOGIC_HERE;

    if (match && !treeNode.isLeaf && !treeNode.children) {
      // Do some loading logic
    }

    return match;
  }}
/>
```

Why can't popup scroll horizontally?

Just turn off virtual scrolling, because the `scrollWidth` of the complete list cannot be accurately measured when virtual scrolling is turned on.