TreeSelect

Tree selection control.

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

The most basic usage.

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

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

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

export default App;
```

Multiple Selection

Multiple selection usage.

```
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' }
          },
        ],
      },
    ],
  },
];
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, overflo
      placeholder="Please select"
      allowClear
      multiple
      treeDefaultExpandAll
      onChange={onChange}
      treeData={treeData}
    />
  );
};

export default App;
```

Generate from tree data

The tree structure can be populated using `treeData` property. This is a quick and easy way to provide the tree content.

```
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, overflo
      treeData={treeData}
      placeholder="Please select"
      treeDefaultExpandAll
      onChange={onChange}
    />
  );
};

export default App;
```

Checkable

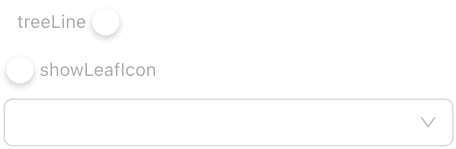
Multiple and checkable.

```
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 ', value);
    setValue(newValue);
  };

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

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

Asynchronous loading

Asynchronous loading tree node.

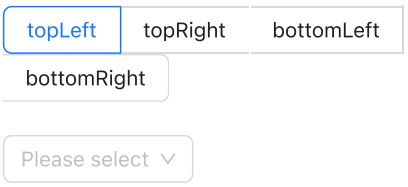
```
import React, { useState } from 'react';
import type { TreeSelectProps } from 'antd';
import { TreeSelect } from 'antd';
import type { DefaultOptionType } from 'antd/e

const App: React.FC = () => {
  const [value, setValue] = useState<string>()
  const [treeData, setTreeData] = useState<Omi
    { id: 1, pId: 0, value: '1', title: 'Expar
    { id: 2, pId: 0, value: '2', title: 'Expar
    { id: 3, pId: 0, value: '3', title: 'Tree
  ]);

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

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

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

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

export default App;
```



Show Tree Line

Use `treeLine` to show the line style.

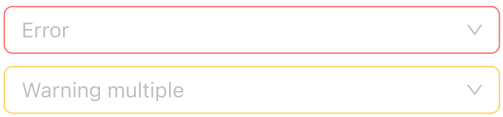
```
import React, { useState } from 'react';
import { Space, Switch, TreeSelect } from 'ant

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: 'sss',
            title: 'sss',
          },
        ],
      },
    ],
  },
];

const App: React.FC = () => {
  const [treeLine, setTreeLine] = useState(tru
  const [showLeafIcon, setShowLeafIcon] = useS

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



Placement

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

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

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' }
          },
        ],
      },
    ],
  },
];
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">botto
        <Radio.Button value="bottomRight">bott
      </Radio.Group>
      <br />
      <br />

      <TreeSelect
        showSearch
        dropdownStyle={{ maxHeight: 400, overf
```



Status

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

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

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

export default App;
```

```
        placeholder="Please select"
        dropdownMatchSelectWidth={false}
        placement={placement}
        allowClear
        treeDefaultExpandAll
        treeData={treeData}
      />
    </>
  );

export default App;
```

Property	Description	Type	Defau
allowClear	Whether allow clear	boolean	false
autoClearSearchValue	If auto clear search input value when multiple select is selected/deselected	boolean	true
bordered	Whether has border style	boolean	true
defaultValue	To set the initial selected treeNode(s)	string | string[]	–
disabled	Disabled or not	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, props) => ReactNode	–
dropdownStyle	To set the style of the dropdown menu	CSSProperties	–
fieldNames	Customize node label, value, children field name	object	{ labe value child: }
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)	funct:
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	boolean	false

Property	Description	Type	Defau
	format of value from `string` to {value: string, label: ReactNode, halfChecked: string[]}		
listHeight	Config popup height	number	256
loadData	Load data asynchronously	function(node)	-
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	-
multiple	Support multiple or not, will be `true` when enable `treeCheckable`	boolean	false
notFoundContent	Specify content to show when no result matches	ReactNode	`Not F`
placeholder	Placeholder of the select input	string	-
placement	The position where the selection box pops up	`bottomLeft` `bottomRight` `topLeft` `topRight`	botto
searchValue	Work with `onSearch` to make search value controlled	string	-
showArrow	Whether to show the `suffixIcon`	boolean	`true`
showCheckedStrategy	The way show selected item in box when `treeCheckable` set. Default: just show child nodes. `TreeSelect.SHOW_ALL`: show all checked treeNodes (include parent treeNode). `TreeSelect.SHOW_PARENT`: show checked treeNodes (just show parent treeNode)	`TreeSelect.SHOW_ALL` | `TreeSelect.SHOW_PARENT` | `TreeSelect.SHOW_CHILD`	`TreeS`
showSearch	Support search or not	boolean	singl multi
size	To set the size of the select input	`large` | `middle` | `small`	-
status	Set validation status	'error' | 'warning'	-

Property	Description	Type	Defau
suffixIcon	The custom suffix icon,you must set `showArrow` to `true` manually in multiple selection mode	ReactNode	-
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	false
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	false
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 providing object to `treeDataSimpleMode`	boolean | object<{ id: string, pId: string, rootPId: string }>	false
treeDefaultExpandAll	Whether to expand all treeNodes by default	boolean	false
treeDefaultExpandedKeys	Default expanded treeNodes	string[]	-
treeExpandAction	Tree title open logic when click, optional: false | `click` | `doubleClick`	string | boolean	false
treeExpandedKeys	Set expanded keys	string[]	-
treeIcon	Shows the icon before a TreeNode's title. There is no default style; you must set a custom	boolean	false

Property	Description	Type	Defau
	style for it if set to `true`		
treeLoadedKeys	(Controlled) Set loaded tree nodes, work with `loadData` only	string[]	[]
treeLine	Show the line. Ref Tree - showLine	boolean | object	false
treeNodeFilterProp	Will be used for filtering if `filterTreeNode` returns true	string	`value`
treeNodeLabelProp	Will render as content of select	string	`title`
value	To set the current selected treeNode(s)	string | string[]	-
virtual	Disable virtual scroll when set to false	boolean	true
onChange	A callback function, can be executed when selected treeNodes or input value change	function(value, label, extra)	-
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)	-

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

▼ 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`
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`
colorPrimaryBorder	The stroke color under the main color gradient, used on the stroke of components such as Slider.	`string`	`#91caff`

Token Name	Description	Type	Default Value
colorPrimaryHover	Hover state under the main color gradient.	`string`	`#4096ff`
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)`
colorTextDisabled	Control the color of text in disabled state.	`string`	`rgba(0, 0, 0, 0.25)`
colorTextLightSolid	Control the highlight color of text with background color, such as the text in Primary Button components.	`string`	`#fff`
colorWhite	Pure white color don't changed by theme	`string`	`#fff`
borderRadius	Border radius of base components	`number`	6
borderRadiusSM	SM size border radius, used in small size components, such as Button, Input, Select and other input components in small size	`number`	4
controlHeightSM	SM component height	`number`	24
controlInteractiveSize	Control the interactive size of control component.	`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)`
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
fontSizeLG	Large font size	`number`	16
lineHeight	Line height of text.	`number`	1.5714285714285714
lineType	Border style of base components	`string`	solid
lineWidth	Border width of base components	`number`	1
lineWidthBold	The default line width of the outline class components, such as Button, Input,	`number`	2

Token Name	Description	Type	Default Value
	Select, etc.		
lineWidthFocus	Control the width of the line when the component is in focus state.	`number`	4
marginXS	Control the margin of an element, with a small size.	`number`	8
motionDurationFast	Motion speed, fast speed. Used for small element animation interaction.	`string`	0.1s
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
motionEaseInBack	Preset motion curve.	`string`	cubic-bezier(0.71, -0.46, 0.88, 0.6)
motionEaseOutBack	Preset motion curve.	`string`	cubic-bezier(0.12, 0.4, 0.29, 1.46)
paddingXS	Control the extra small padding of the element.	`number`	8

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/wk080nn81k

Why sometime customize Option cause scroll break?

You can ref Select FAQ.