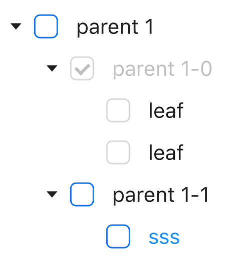
Tree

A hierarchical list structure component.

When To Use

Almost anything can be represented in a tree structure. Examples include directories, organization hierarchies, biological classifications, countries, etc. The `Tree` component is a way of representing the hierarchical relationship between these things. You can also expand, collapse, and select a treeNode within a `Tree`.

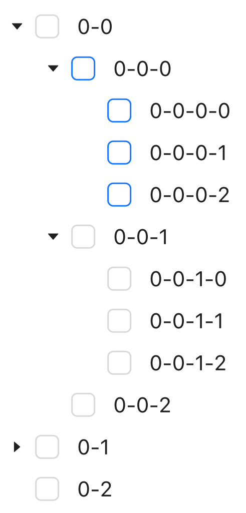
Examples



Basic

The most basic usage, tell you how to use checkable, selectable, disabled, defaultExpandKeys, and etc.

```
import React from 'react';
import { Tree } from 'antd';
import type { DataNode, TreeProps } from 'ant

const treeData: DataNode[] = [
  {
    title: 'parent 1',
    key: '0-0',
    children: [
      {
        title: 'parent 1-0',
        key: '0-0-0',
        disabled: true,
        children: [
          {
            title: 'leaf',
            key: '0-0-0-0',
            disableCheckbox: true,
          },
          {
            title: 'leaf',
            key: '0-0-0-1',
          },
        ],
      },
      {
        title: 'parent 1-1',
        key: '0-0-1',
        children: [{ title: <span style={{ col
      },
    ],
  },
];

const App: React.FC = () => {
  const onSelect: TreeProps['onSelect'] = (sel
    console.log('selected', selectedKeys, inf
  };

  const onCheck: TreeProps['onCheck'] = (chec
    console.log('onCheck', checkedKeys, info);
  };

  return (
    <Tree
      checkable
      defaultExpandedKeys={['0-0-0', '0-0-1']}
      defaultSelectedKeys={['0-0-0', '0-0-1']}
      defaultCheckedKeys={['0-0-0', '0-0-1']}
      onSelect={onSelect}
      onCheck={onCheck}
      treeData={treeData}
    />
  );
```



Controlled Tree

Controlled mode lets parent nodes reflect the status of child nodes more intelligently.

```
import React, { useState } from 'react';
import { Tree } from 'antd';
import type { DataNode } from 'antd/es/tree';

const treeData: DataNode[] = [
  {
    title: '0-0',
    key: '0-0',
    children: [
      {
        title: '0-0-0',
        key: '0-0-0',
        children: [
          { title: '0-0-0-0', key: '0-0-0-0' }
          { title: '0-0-0-1', key: '0-0-0-1' }
          { title: '0-0-0-2', key: '0-0-0-2' }
        ],
      },
      {
        title: '0-0-1',
        key: '0-0-1',
        children: [
          { title: '0-0-1-0', key: '0-0-1-0' }
          { title: '0-0-1-1', key: '0-0-1-1' }
          { title: '0-0-1-2', key: '0-0-1-2' }
        ],
      },
      {
        title: '0-0-2',
        key: '0-0-2',
      },
    ],
  },
  {
    title: '0-1',
    key: '0-1',
    children: [
      { title: '0-1-0-0', key: '0-1-0-0' },
      { title: '0-1-0-1', key: '0-1-0-1' },
      { title: '0-1-0-2', key: '0-1-0-2' },
    ],
  },
  {
    title: '0-2',
    key: '0-2',
  },
];
```

- 0-0
  - 0-0-0
    - 0-0-0-0
    - 0-0-0-1
    - 0-0-0-2
  - 0-0-1
  - 0-0-2
- 0-1
- 0-2

draggable

Drag treeNode to insert after the other treeNode or insert into the other parent TreeNode.

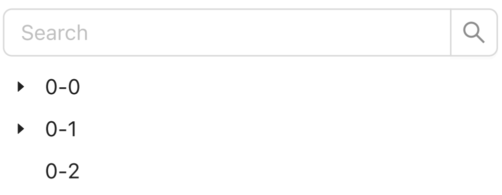
```
import React, { useState } from 'react';
import { Tree } from 'antd';
import type { DataNode, TreeProps } from 'antc

const x = 3;
const y = 2;
const z = 1;
const defaultData: DataNode[] = [];

const generateData = (_level: number, _preKey?
  const preKey = _preKey || '0';
  const tns = _tns || defaultData;

  const children: React.Key[] = [];
  for (let i = 0; i < x; i++) {
    const key = `${preKey}-${i}`;
    tns.push({ title: key, key });
    if (i < y) {
      children.push(key);
    }
  }
  if (_level < 0) {
    return tns;
  }
  const level = _level - 1;
  children.forEach((key, index) => {
    tns[index].children = [];
    return generateData(level, key, tns[index
  });
};
generateData(z);

const App: React.FC = () => {
  const [gData, setGData] = useState(defaultD
  const [expandedKeys] = useState(['0-0', '0-

  const onDragEnter: TreeProps['onDragEnter']
    console.log(info);
    // expandedKeys 需要受控时设置
    // setExpandedKeys(info.expandedKeys)
  };

  const onDrop: TreeProps['onDrop'] = (info) =
    console.log(info);
    const dropKey = info.node.key;
    const dragKey = info.dragNode.key;
    const dropPos = info.node.pos.split('-');
    const dropPosition = info.dropPosition - N

    const loop = (
      data: DataNode[],
```

- Expand to load
- Expand to load
- Tree Node

load data asynchronously

To load data asynchronously when click to expand a treeNode.

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

interface DataNode {
  title: string;
  key: string;
  isLeaf?: boolean;
  children?: DataNode[];
}

const initTreeData: DataNode[] = [
  { title: 'Expand to load', key: '0' },
  { title: 'Expand to load', key: '1' },
  { title: 'Tree Node', key: '2', isLeaf: true
];

// It's just a simple demo. You can use tree m
const updateTreeData = (list: DataNode[], key:
  list.map((node) => {
    if (node.key === key) {
      return {
        ...node,
        children,
      };
    }
    if (node.children) {
      return {
        ...node,
        children: updateTreeData(node.childre
      };
    }
    return node;
  });

const App: React.FC = () => {
  const [treeData, setTreeData] = useState(ini

  const onLoadData = ({ key, children }: any)
    new Promise<void>((resolve) => {
      if (children) {
        resolve();
        return;
      }
      setTimeout(() => {
        setTreeData((origin) =>
          updateTreeData(origin, key, [
            { title: 'Child Node', key: `${key
            { title: 'Child Node', key: `${key
          ]),
        );

        resolve();
      }, 1000);
    });

  return <Tree loadData={onLoadData} treeData=
};

export default App;
```


Search

0-0

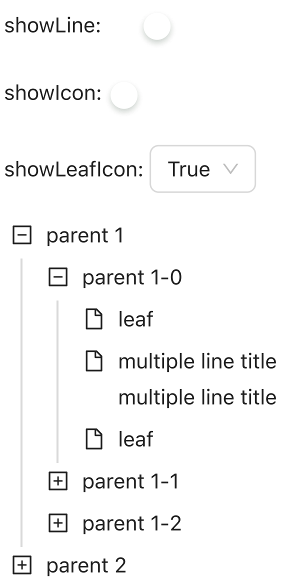
0-1

0-2

Searchable

Searchable Tree.

```
import React, { useMemo, useState } from 'reac
import { Input, Tree } from 'antd';
import type { DataNode } from 'antd/es/tree';

const { Search } = Input;

const x = 3;
const y = 2;
const z = 1;
const defaultData: DataNode[] = [];

const generateData = (_level: number, _preKey?
  const preKey = _preKey || '0';
  const tns = _tns || defaultData;

  const children: React.Key[] = [];
  for (let i = 0; i < x; i++) {
    const key = `${preKey}-${i}`;
    tns.push({ title: key, key });
    if (i < y) {
      children.push(key);
    }
  }
  if (_level < 0) {
    return tns;
  }
  const level = _level - 1;
  children.forEach((key, index) => {
    tns[index].children = [];
    return generateData(level, key, tns[index]
  });
};
generateData(z);

const dataList: { key: React.Key; title: strin
const generateList = (data: DataNode[]) => {
  for (let i = 0; i < data.length; i++) {
    const node = data[i];
    const { key } = node;
    dataList.push({ key, title: key as string
    if (node.children) {
      generateList(node.children);
    }
  }
};
generateList(defaultData);

const getParentKey = (key: React.Key, tree: Dc
  let parentKey: React.Key;
  for (let i = 0; i < tree.length; i++) {
    const node = tree[i];
    if (node.children) {
      if (node.children.some((item) => item.ke
        parentKey = node.key;
      } else if (getParentKey(key, node.childr
        parentKey = getParentKey(key, node.chi
      }
    }
  }
```


showLine:

showIcon:

showLeafIcon: True

parent 1

parent 1-0

leaf

multiple line title
multiple line title

leaf

parent 1-1

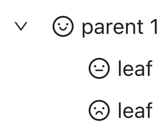
parent 1-2

parent 2

Tree with line

Tree with connected line between nodes, turn on by `showLine`, customize the preset icon by `switcherIcon`.

```
import React, { useState } from 'react';
import { CarryOutOutlined, CheckOutlined, Form
import { Select, Switch, Tree } from 'antd';
import type { DataNode } from 'antd/es/tree';

const treeData: DataNode[] = [
  {
    title: 'parent 1',
    key: '0-0',
    icon: <CarryOutOutlined />,
    children: [
      {
        title: 'parent 1-0',
        key: '0-0-0',
        icon: <CarryOutOutlined />,
        children: [
          { title: 'leaf', key: '0-0-0-0', icc
          {
            title: (
              <>
                <div>multiple line title</div>
                <div>multiple line title</div>
              </>
            ),
            key: '0-0-0-1',
            icon: <CarryOutOutlined />,
          },
          { title: 'leaf', key: '0-0-0-2', icc
        ],
      },
      {
        title: 'parent 1-1',
        key: '0-0-1',
        icon: <CarryOutOutlined />,
        children: [{ title: 'leaf', key: '0-0-
      },
      {
        title: 'parent 1-2',
        key: '0-0-2',
        icon: <CarryOutOutlined />,
        children: [
          { title: 'leaf', key: '0-0-2-0', icc
          {
```



parent 1

leaf

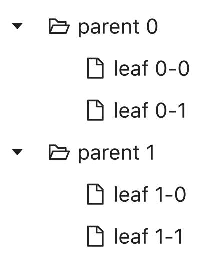
leaf

Customize Icon

You can customize icons for different nodes.

```
import React from 'react';
import {
  DownOutlined,
  FrownFilled,
  FrownOutlined,
  MehOutlined,
  SmileOutlined,
} from '@ant-design/icons';
import { Tree } from 'antd';
import type { DataNode } from 'antd/es/tree';

const treeData: DataNode[] = [
  {
    title: 'parent 1',
    key: '0-0',
    icon: <SmileOutlined />,
    children: [
      {
        title: 'leaf',
        key: '0-0-0',
        icon: <MehOutlined />,
      },
      {
        title: 'leaf',
        key: '0-0-1',
        icon: ({ selected }) => (selected ? <F
      },
    ],
  },
];

const App: React.FC = () => (
  <Tree
    showIcon
    defaultExpandAll
    defaultSelectedKeys={['0-0-0']}
    switcherIcon={<DownOutlined />}
    treeData={treeData}
  />
);

export default App;
```

```
  }, [searchValue]);

  return (
    <div>
      <Search style={{ marginBottom: 8 }} plac
      <Tree
        onExpand={onExpand}
        expandedKeys={expandedKeys}
        autoExpandParent={autoExpandParent}
        treeData={treeData}
      />
    </div>
  );
};

export default App;
```



parent 0

leaf 0-0

leaf 0-1

parent 1

leaf 1-0

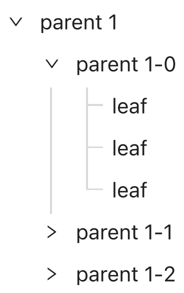
leaf 1-1

directory

Built-in directory tree. `multiple` support `ctrl(Windows)` / `command(Mac)` selection.

```
import React from 'react';
import { Tree } from 'antd';
import type { DataNode, DirectoryTreeProps } f

const { DirectoryTree } = Tree;

const treeData: DataNode[] = [
  {
    title: 'parent 0',
    key: '0-0',
    children: [
      { title: 'leaf 0-0', key: '0-0-0', isLe
      { title: 'leaf 0-1', key: '0-0-1', isLe
    ],
  },
  {
    title: 'parent 1',
    key: '0-1',
    children: [
      { title: 'leaf 1-0', key: '0-1-0', isLe
      { title: 'leaf 1-1', key: '0-1-1', isLe
    ],
  },
];

const App: React.FC = () => {
  const onSelect: DirectoryTreeProps['onSelect
    console.log('Trigger Select', keys, info);
  };

  const onExpand: DirectoryTreeProps['onExpan
    console.log('Trigger Expand', keys, info);
  };

  return (
    <DirectoryTree
      multiple
      defaultExpandAll
      onSelect={onSelect}
      onExpand={onExpand}
      treeData={treeData}
    />
  );
};

export default App;
```

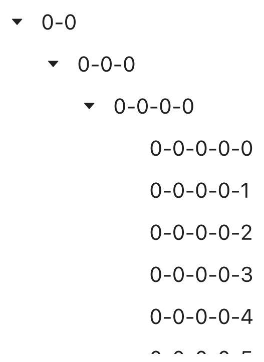
```
        showLine={showLine ? { showLeafIcon }
        showIcon={showIcon}
        defaultExpandedKeys={['0-0-0']}
        onSelect={onSelect}
        treeData={treeData}
      />
    </div>
  );
};
```


parent 1
parent 1-0
leaf
leaf
leaf
parent 1-1
parent 1-2

Customize collapse/expand icon

customize collapse/expand icon of tree node

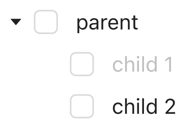
```
import React from 'react';
import { DownOutlined } from '@ant-design/ico
import { Tree } from 'antd';
import type { DataNode, TreeProps } from 'ant

const treeData: DataNode[] = [
  {
    title: 'parent 1',
    key: '0-0',
    children: [
      {
        title: 'parent 1-0',
        key: '0-0-0',
        children: [
          {
            title: 'leaf',
            key: '0-0-0-0',
          },
          {
            title: 'leaf',
            key: '0-0-0-1',
          },
          {
            title: 'leaf',
            key: '0-0-0-2',
          },
        ],
      },
      {
        title: 'parent 1-1',
        key: '0-0-1',
        children: [
          {
            title: 'leaf',
            key: '0-0-1-0',
          },
        ],
      },
      {
        title: 'parent 1-2',
        key: '0-0-2',
        children: [
          {
            title: 'leaf',
            key: '0-0-2-0',
          },
          {
            title: 'leaf',
            key: '0-0-2-1',
          },
        ],
      },
    ],
  },
];
```


0-0
0-0-0
0-0-0-0
0-0-0-0-0
0-0-0-0-1
0-0-0-0-2
0-0-0-0-3
0-0-0-0-4

Virtual scroll

Use virtual list through `height` prop.

```
import React from 'react';
import { Tree } from 'antd';
import type { DataNode } from 'antd/es/tree';

const dig = (path = '0', level = 3) => {
  const list = [];
  for (let i = 0; i < 10; i += 1) {
    const key = `${path}-${i}`;
    const treeNode: DataNode = {
      title: key,
      key,
    };

    if (level > 0) {
      treeNode.children = dig(key, level - 1);
    }

    list.push(treeNode);
  }
  return list;
};

const treeData = dig();

const App: React.FC = () => <Tree treeData={tr

export default App;
```



Block Node

```
import React from 'react';
import { Tree } from 'antd';
import type { DataNode } from 'antd/es/tree';

const treeData: DataNode[] = [
  {
    title: 'parent',
    key: '0',
    children: [
      {
        title: 'child 1',
        key: '0-0',
        disabled: true,
      },
      {
        title: 'child 2',
        key: '0-1',
        disableCheckbox: true,
      },
    ],
  },
];

const App: React.FC = () => (
  <Tree checkable defaultSelectedKeys={['0-1'
);

export default App;
```

API

Tree props

Property	Description	Type	Default
allowDrop	Whether to allow dropping on the node	({ dropNode, dropPosition }) => boolean	-
autoExpandParent	Whether to automatically expand a parent treeNode	boolean	false
blockNode	Whether treeNode fill remaining horizontal space	boolean	false
checkable	Add a Checkbox before the treeNodes	boolean	false

Property	Description	Type	Default
checkedKeys	(Controlled) Specifies the keys of the checked treeNodes (PS: When this specifies the key of a treeNode which is also a parent treeNode, all the children treeNodes of will be checked; and vice versa, when it specifies the key of a treeNode which is a child treeNode, its parent treeNode will also be checked. When `checkable` and `checkStrictly` is true, its object has `checked` and `halfChecked` property. Regardless of whether the child or parent treeNode is checked, they won't impact each other	`string[] | {checked: string[], halfChecked: string[]}`	[]
checkStrictly	Check treeNode precisely; parent treeNode and children treeNodes are not associated	`boolean`	false
defaultCheckedKeys	Specifies the keys of the default checked treeNodes	`string[]`	[]
defaultExpandAll	Whether to expand all treeNodes by default	`boolean`	false
defaultExpandedKeys	Specify the keys of the default expanded treeNodes	`string[]`	[]
defaultExpandParent	If auto expand parent treeNodes when init	`boolean`	true
defaultSelectedKeys	Specifies the keys of the default selected treeNodes	`string[]`	[]
disabled	Whether disabled the tree	`boolean`	false
draggable	Specifies whether this Tree or the node is draggable. Use `icon: false` to	`boolean | ((node: DataNode) => boolean) | { icon?: React.ReactNode | false, nodeDraggable?:`	false

Property	Description	Type	Default
	disable drag handler icon	(node: DataNode) => boolean }	
expandedKeys	(Controlled) Specifies the keys of the expanded treeNodes	string[]	[]
fieldNames	Customize node title, key, children field name	object	{ title: `title`, key: `key`, children: `children` }
filterTreeNode	Defines a function to filter (highlight) treeNodes. When the function returns `true`, the corresponding treeNode will be highlighted	function(node)	-
height	Config virtual scroll height. Will not support horizontal scroll when enable this	number	-
icon	Customize treeNode icon	ReactNode | (props) => ReactNode	-
loadData	Load data asynchronously	function(node)	-
loadedKeys	(Controlled) Set loaded tree nodes. Need work with `loadData`	string[]	[]
multiple	Allows selecting multiple treeNodes	boolean	false
rootClassName	ClassName on the root element	string	-
rootStyle	Style on the root element	CSSProperties	-
selectable	Whether can be selected	boolean	true
selectedKeys	(Controlled) Specifies the keys of the selected treeNodes	string[]	-
showIcon	Shows the icon before a TreeNode's title. There is no default style; you must set a custom	boolean	false

Property	Description	Type	Default
	style for it if set to true		
showLine	Shows a connecting line	boolean | {showLeafIcon: boolean | ReactNode | ((props: AntTreeNodeProps) => ReactNode)}	false
switcherIcon	Customize collapse/expand icon of tree node	ReactNode | ((props: AntTreeNodeProps) => ReactNode)	-
titleRender	Customize tree node title render	(nodeData) => ReactNode	-
treeData	The treeNodes data Array, if set it then you need not to construct children TreeNode. (key should be unique across the whole array)	array<{ key, title, children, [disabled, selectable] }>	-
virtual	Disable virtual scroll when set to false	boolean	true
onCheck	Callback function for when the onCheck event occurs	function(checkedKeys, e:{checked: bool, checkedNodes, node, event, halfCheckedKeys})	-
onDragEnd	Callback function for when the onDragEnd event occurs	function({event, node})	-
onDragEnter	Callback function for when the onDragEnter event occurs	function({event, node, expandedKeys})	-
onDragLeave	Callback function for when the onDragLeave event occurs	function({event, node})	-
onDragOver	Callback function for when the onDragOver event occurs	function({event, node})	-
onDragStart	Callback function for when the onDragStart event occurs	function({event, node})	-
onDrop	Callback function for when the onDrop event occurs	function({event, node, dragNode, dragNodesKeys})	-

Property	Description	Type	Default
onExpand	Callback function for when a treeNode is expanded or collapsed	function(expandedKeys, {expanded: bool, node})	-
onLoad	Callback function for when a treeNode is loaded	function(loadedKeys, {event, node})	-
onRightClick	Callback function for when the user right clicks a treeNode	function({event, node})	-
onSelect	Callback function for when the user clicks a treeNode	function(selectedKeys, e:{selected: bool, selectedNodes, node, event})	-

TreeNode props

Property	Description	Type	Default
checkable	When Tree is checkable, set TreeNode display Checkbox or not	boolean	-
disableCheckbox	Disables the checkbox of the treeNode	boolean	false
disabled	Disables the treeNode	boolean	false
icon	Customize icon. When you pass component, whose render will receive full TreeNode props as component props	ReactNode | (props) => ReactNode	-
isLeaf	Determines if this is a leaf node(effective when `loadData` is specified). `false` will force trade TreeNode as a parent node	boolean	-
key	Used with (default)ExpandedKeys / (default)CheckedKeys / (default)SelectedKeys. P.S.: It must be unique in all of treeNodes of the tree	string	(internal calculated position of treeNode)
selectable	Set whether the treeNode can be selected	boolean	true
title	Title	ReactNode	`---`

DirectoryTree props

Property	Description	Type	Default
expandAction	Directory open logic, optional: false | `click` | `doubleClick`	string | boolean	`click`

Note

Before `3.4.0`: The number of treeNodes can be very large, but when `checkable=true`, it will increase the compute time. So, we cache some calculations (e.g. `this.treeNodesStates`) to avoid double computing. But, this brings some restrictions. When you load treeNodes asynchronously, you should render tree like this:

```
{
  this.state.treeData.length ? (
    <Tree>
      {this.state.treeData.map((data) => (
        <TreeNode />
      ))}
    </Tree>
  ) : (
    'loading tree'
  );
}
```

Tree Methods

Name	D
`scrollTo({ key: string | number; align?: 'top' | 'bottom' | 'auto'; offset?: number })`	S i

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)
colorBorder	Default border color, used to separate different elements, such as: form separator, card separator, etc.	`string`	#d9d9d9
colorError	Used to represent the visual elements of the operation failure, such as the error Button, error Result component, etc.	`string`	#ff4d4f
colorLinkHover	Control the color of hyperlink when hovering.	`string`	#69b1ff
colorSplit	Used as the color of separator, this color is the same as colorBorderSecondary but	`string`	rgba(5, 5, 5, 0.06)

Token Name	Description	Type	Default Value
	with transparency.		
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)`
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`	□ `#faad14`
borderRadiusLG	LG size border radius, used in some large border radius components, such as Card, Modal and other components.	`number`	8
controlHeight	The height of the basic controls such as buttons and input boxes in Ant Design	`number`	32
controlHeightLG	LG component height	`number`	40
controlItemBgActive	Control the background color of control component item when active.	`string`	□ `#e6f4ff`
controlItemBgActiveHover	Control the background color of control component item when hovering and active.	`string`	□ `#bae0ff`
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
fontSizeIcon	Control the font size of operation icon in Select, Cascader, etc. Normally same as fontSizeSM.	`number`	12
lineHeight	Line height of text.	`number`	1.5714285714285714
lineType	Border style of base components	`string`	solid
lineWidth	Border width of base components	`number`	1
margin	Control the margin of an element, with a medium size.	`number`	16

Token Name	Description	Type	Default Value
marginXS	Control the margin of an element, with a small size.	`number`	8
marginXXS	Control the margin of an element, with the smallest size.	`number`	4
motionDurationSlow	Motion speed, slow speed. Used for large element animation interaction.	`string`	0.3s
paddingSM	Control the small padding of the element.	`number`	12
paddingXS	Control the extra small padding of the element.	`number`	8

FAQ

How to hide file icon when use showLine?

File icon realize by using switcherIcon. You can overwrite the style to hide it: https://codesandbox.io/s/883vo47xp8

Why defaultExpandAll not working on ajax data?

`default` prefix prop only works when initializing. So `defaultExpandAll` has already executed when ajax load data. You can control `expandedKeys` or render Tree when data loaded to realize expanded all.

Virtual scroll limitation

Virtual scroll only render items in visible region. Thus not support auto width (like long `title` with horizontal scroll).

What does `disabled` node work logic in the tree?

Tree change its data by conduction. Includes checked or auto expanded, it will conduction state to parent / children node until current node is `disabled`. So if a controlled node is `disabled`, it will only modify self state and not affect other nodes. For example, a parent node contains 3 child nodes and one of them is `disabled`. When check the parent node, it will only check rest 2 child nodes. As the same, when check these 2 child node, parent will be checked whatever checked state the `disabled` one is.

This conduction logic prevent that modify `disabled` parent checked state by check children node and user can not modify directly with click parent which makes the interactive conflict. If you want to modify this conduction logic, you can customize it with `checkStrictly` prop.