何时使用 {#when-to-use}

类似 Select 的选择控件，可选择的数据结构是一个树形结构时，可以使用 TreeSelect，例如公司层级、学科系统、分类目录等等。

代码演示

基本

```
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;
```

多选

```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;
```

从数据直接生成

```
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;
```

可勾选

```
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;
```

异步加载

```
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;
```

线性样式

```
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;
```

弹出位置

```
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;
```

形态变体

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;
```

自定义状态

```
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;
```

最大选中数量

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;
```

前后缀

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;
```

组件 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

通用属性参考：通用属性

Tree props

参数	说明	类型
allowClear	自定义清除按钮	boolean | { clearIcon?: ReactNode }
autoClearSearchValue	当多选模式下值被选择，自动清空搜索框	boolean
defaultOpen	是否默认展开下拉菜单	boolean
defaultValue	指定默认选中的条目	string | string[]
disabled	是否禁用	boolean
popupClassName	下拉菜单的 className 属性	string

popupMatchSelectWidth	下拉菜单和选择器同宽。默认将设置 `min-width`，当值小于选择框宽度时会被忽略。false 时会关闭虚拟滚动	boolean | number	
dropdownRender	自定义下拉框内容	(originNode: ReactNode, props) => ReactNode	
dropdownStyle	下拉菜单的样式	object	
fieldNames	自定义节点 label、value、children 的字段	object	
filterTreeNode	是否根据输入项进行筛选，默认用 treeNodeFilterProp 的值作为要筛选的 TreeNode 的属性值	boolean | function(inputValue: string, treeNode: TreeNode) (函数需要返回 bool 值)	
getPopupContainer	菜单渲染父节点。默认渲染到 body 上，如果你遇到菜单滚动定位问题，试试修改为滚动的区域，并相对其定位。[示例]	function(triggerNode)	
labelInValue	是否把每个选项的 label 包装到 value 中，会把 value 类型从 `string` 变为 {value: string, label: ReactNode, halfChecked(treeCheckStrictly 时有效): string[] } 的格式	boolean	
listHeight	设置弹窗滚动高度	number	
loadData	异步加载数据。在过滤时不会调用以防止网络堵塞，可参考 FAQ 获得更多内容	function(node)	
maxCount	指定可选中的最多 items 数量，仅在 `multiple=true` 时生效。如果此时 (`showCheckedStrategy = 'SHOW_ALL'` 且未开启 `treeCheckStrictly`)，或使用 `showCheckedStrategy = 'SHOW_PARENT'`，则maxCount 无效。	number	
maxTagCount	最多显示多少个 tag，响应式模式会对性能产生损耗	number | `responsive`	

maxTagPlaceholder	隐藏 tag 时显示的内容	ReactNode | function(omittedValues)	
maxTagTextLength	最大显示的 tag 文本长度	number	
multiple	支持多选（当设置 treeCheckable 时自动变为 true）	boolean	
notFoundContent	当下拉列表为空时显示的内容	ReactNode	
open	是否展开下拉菜单	boolean	
placeholder	选择框默认文字	string	
placement	选择框弹出的位置	`bottomLeft` `bottomRight` `topLeft` `topRight`	
prefix	自定义前缀	ReactNode	
searchValue	搜索框的值，可以通过 `onSearch` 获取用户输入	string	
showCheckedStrategy	配置 `treeCheckable` 时，定义选中项回填的方式。`TreeSelect.SHOW_ALL`: 显示所有选中节点(包括父节点)。`TreeSelect.SHOW_PARENT`: 只显示父节点(当父节点下所有子节点都选中时)。 默认只显示子节点	`TreeSelect.SHOW_ALL` | `TreeSelect.SHOW_PARENT` | `TreeSelect.SHOW_CHILD`	
showSearch	是否支持搜索框	boolean	
size	选择框大小	`large` | `middle` | `small`	
status	设置校验状态	'error' | 'warning'	
suffixIcon	自定义的选择框后缀图标	ReactNode	
switcherIcon	自定义树节点的展开/折叠图标	ReactNode | ((props: AntTreeNodeProps) => ReactNode)	
tagRender	自定义 tag 内容，多选时生效	(props) => ReactNode	
treeCheckable	显示 Checkbox	boolean	

treeCheckStrictly	`checkable` 状态下节点选择完全受控（父子节点选中状态不再关联），会使得 `labelInValue` 强制为 true	boolean	
treeData	treeNodes 数据，如果设置则不需要手动构造 TreeNode 节点（value 在整个树范围内唯一）	array<{value, title, children, [disabled, disableCheckbox, selectable, checkable]}>	
treeDataSimpleMode	使用简单格式的 treeData，具体设置参考可设置的类型 (此时 treeData 应变为这样的数据结构: [{id:1, pId:0, value:'1', title:"test1",...},...]， `pId` 是父节点的 id)	boolean | object<{ id: string, pId: string, rootPId: string }>	
treeTitleRender	自定义渲染节点	(nodeData) => ReactNode	
treeDefaultExpandAll	默认展开所有树节点	boolean	
treeDefaultExpandedKeys	默认展开的树节点	string[]	
treeExpandAction	点击节点 title 时的展开逻辑，可选：false | `click` | `doubleClick`	string | boolean	
treeExpandedKeys	设置展开的树节点	string[]	
treeIcon	是否展示 TreeNode title 前的图标，没有默认样式，如设置为 true，需要自行定义图标相关样式	boolean	
treeLine	是否展示线条样式，请参考 Tree - showLine	boolean | object	
treeLoadedKeys	（受控）已经加载的节点，需要配合 `loadData` 使用	string[]	
treeNodeFilterProp	输入项过滤对应的 treeNode 属性	string	
treeNodeLabelProp	作为显示的 prop 设置	string	
value	指定当前选中的条目	string | string[]	
variant	形态变体	`outlined` | `borderless` | `filled` | `underlined`	

virtual	设置 false 时关闭虚拟滚动	boolean	
onChange	选中树节点时调用此函数	function(value, label, extra)	
onDropdownVisibleChange	展开下拉菜单的回调	function(open)	
onSearch	文本框值变化时的回调	function(value: string)	
onSelect	被选中时调用	function(value, node, extra)	
onTreeExpand	展示节点时调用	function(expandedKeys)	
onPopupScroll	下拉列表滚动时的回调	(event: UIEvent) => void	

Tree 方法

名称	描述	版本
blur()	移除焦点	
focus()	获取焦点	

TreeNode props

> *建议使用 treeData 来代替 TreeNode，免去手动构造的麻烦*

参数	说明	类型	默认值	版本
checkable	当树为 Checkbox 时，设置独立节点是否展示 Checkbox	boolean	-	
disableCheckbox	禁掉 Checkbox	boolean	false	
disabled	是否禁用	boolean	false	
isLeaf	是否是叶子节点	boolean	false	
key	此项必须设置（其值在整个树范围内唯一）	string	-	
selectable	是否可选	boolean	true	
title	树节点显示的内容	ReactNode	–––	
value	默认根据此属性值进行筛选（其值在整个树范围内唯一）	string	-	

主题变量（Design Token）

FAQ

onChange 时如何获得父节点信息?

从性能角度考虑，我们默认不透出父节点信息。你可以这样获得：https://codesandbox.io/s/get-parent-node-in-onchange-eb1608

自定义 Option 样式导致滚动异常怎么办?

请参考 Select 的 FAQ。

为何在搜索时 `loadData` 不会触发展开?

在 v4 alpha 版本中，默认在搜索时亦会进行搜索。但是经反馈，在输入时会快速阻塞网络。因而改为搜索不触发 `loadData` 。但是你仍然可以通过 `filterTreeNode` 处理异步加载逻辑：

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

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

    return match;
  }}
/>
```

为何弹出框不能横向滚动?

关闭虚拟滚动即可，因为开启虚拟滚动时无法准确的测量完整列表的 `scrollWidth` 。