何时使用 {#when-to-use}

- 需要一个输入框而不是选择器。
- 需要输入建议/辅助提示。

和 Select 的区别是：

- AutoComplete 是一个带提示的文本输入框，用户可以自由输入，关键词是辅助**输入**。
- Select 是在限定的可选项中进行选择，关键词是**选择**。

代码演示

基本使用

```
import React, { useState } from 'react';
import { AutoComplete } from 'antd';
import type { AutoCompleteProps } from 'antd';

const mockVal = (str: string, repeat = 1) => ({
  value: str.repeat(repeat),
});

const App: React.FC = () => {
  const [value, setValue] = useState('');
  const [options, setOptions] = useState<AutoCompleteProps['options']>([]);
  const [anotherOptions, setAnotherOptions] =
useState<AutoCompleteProps['options']>([]);

  const getPanelValue = (searchText: string) =>
    !searchText ? [] : [mockVal(searchText), mockVal(searchText, 2),
mockVal(searchText, 3)];

  const onSelect = (data: string) => {
    console.log('onSelect', data);
  };

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

  return (
    <>
      <AutoComplete
        options={options}
        style={{ width: 200 }}
        onSelect={onSelect}
        onSearch={(text) => setOptions(getPanelValue(text))}
        placeholder="input here"
      />
      <br />
      <br />
      <AutoComplete
        value={value}
        options={anotherOptions}
        style={{ width: 200 }}
        onSelect={onSelect}
        onSearch={(text) => setAnotherOptions(getPanelValue(text))}
        onChange={onChange}
        placeholder="control mode"
      />
    </>
  );
};

export default App;
```

自定义选项

```
import React from 'react';
import { AutoComplete } from 'antd';
import type { AutoCompleteProps } from 'antd';

const App: React.FC = () => {
  const [options, setOptions] =
React.useState<AutoCompleteProps['options']>([]);
  const handleSearch = (value: string) => {
    setOptions(() => {
      if (!value || value.includes('@')) {
        return [];
      }
      return ['gmail.com', '163.com', 'qq.com'].map((domain) => ({
        label: `${value}@${domain}`,
        value: `${value}@${domain}`,
      }));
    });
  };
  return (
    <AutoComplete
      style={{ width: 200 }}
      onSearch={handleSearch}
      placeholder="input here"
      options={options}
    />
  );
};

export default App;
```

自定义输入组件

```
import React, { useState } from 'react';
import { AutoComplete, Input } from 'antd';
import type { AutoCompleteProps } from 'antd';

const { TextArea } = Input;

const App: React.FC = () => {
  const [options, setOptions] = useState<AutoCompleteProps['options']>([]);

  const handleSearch = (value: string) => {
    setOptions(
      !value ? [] : [{ value }, { value: value + value }, { value: value +
value + value }],
    );
  };

  const handleKeyPress = (ev: React.KeyboardEvent<HTMLTextAreaElement>) =>
{
    console.log('handleKeyPress', ev);
  };

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

  return (
    <AutoComplete
      options={options}
      style={{ width: 200 }}
      onSelect={onSelect}
      onSearch={handleSearch}
    >
      <TextArea
        placeholder="input here"
        className="custom"
        style={{ height: 50 }}
        onKeyPress={handleKeyPress}
      />
    </AutoComplete>
  );
};

export default App;
```

不区分大小写

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

const options = [
  { value: 'Burns Bay Road' },
  { value: 'Downing Street' },
  { value: 'Wall Street' },
];

const App: React.FC = () => (
  <AutoComplete
    style={{ width: 200 }}
    options={options}
    placeholder="try to type `b`"
    filterOption={(inputValue, option) =>
      option!.value.toUpperCase().indexOf(inputValue.toUpperCase()) !== -1
    }
  />
);

export default App;
```

查询模式 - 确定类目

```
import React from 'react';
import { UserOutlined } from '@ant-design/icons';
import { AutoComplete, Flex, Input } from 'antd';

const Title: React.FC<Readonly<{ title?: string }>> = (props) => (
  <Flex align="center" justify="space-between">
    {props.title}
    <a href="https://www.google.com/search?q=antd" target="_blank"
rel="noopener noreferrer">
      more
    </a>
  </Flex>
);

const renderItem = (title: string, count: number) => ({
  value: title,
  label: (
    <Flex align="center" justify="space-between">
      {title}
      <span>
        <UserOutlined /> {count}
      </span>
    </Flex>
  ),
});

const options = [
  {
    label: <Title title="Libraries" />,
    options: [renderItem('AntDesign', 10000), renderItem('AntDesign UI',
10600)],
  },
  {
    label: <Title title="Solutions" />,
    options: [renderItem('AntDesign UI FAQ', 60100), renderItem('AntDesign
FAQ', 30010)],
  },
  {
    label: <Title title="Articles" />,
    options: [renderItem('AntDesign design language', 100000)],
  },
];

const App: React.FC = () => (
  <AutoComplete
    popupClassName="certain-category-search-dropdown"
    popupMatchSelectWidth={500}
    style={{ width: 250 }}
    options={options}
    size="large"
  >
    <Input.Search size="large" placeholder="input here" />
  </AutoComplete>
);

export default App;
```

查询模式 - 不确定类目

```
import React, { useState } from 'react';
import { AutoComplete, Input } from 'antd';
import type { AutoCompleteProps } from 'antd';

const getRandomInt = (max: number, min = 0) => Math.floor(Math.random() *
(max - min + 1)) + min;

const searchResult = (query: string) =>
  Array.from({ length: getRandomInt(5) })
    .join('.')
    .split('.')
    .map((_, idx) => {
      const category = `${query}${idx}`;
      return {
        value: category,
        label: (
          <div
            style={{
              display: 'flex',
              justifyContent: 'space-between',
            }}
          >
            <span>
              Found {query} on{' '}
              <a
                href={`https://s.taobao.com/search?q=${query}`}
                target="_blank"
                rel="noopener noreferrer"
              >
                {category}
              </a>
            </span>
            <span>{getRandomInt(200, 100)} results</span>
          </div>
        ),
      };
    });

const App: React.FC = () => {
  const [options, setOptions] = useState<AutoCompleteProps['options']>([]);

  const handleSearch = (value: string) => {
    setOptions(value ? searchResult(value) : []);
  };

  const onSelect = (value: string) => {
    console.log('onSelect', value);
  };
  return (
    <AutoComplete
      popupMatchSelectWidth={252}
      style={{ width: 300 }}
      options={options}
      onSelect={onSelect}
      onSearch={handleSearch}
      size="large"
    >
      <Input.Search size="large" placeholder="input here" enterButton />
    </AutoComplete>
  );
};

export default App;
```

自定义状态

```
import React, { useState } from 'react';
import { AutoComplete, Space } from 'antd';
import type { AutoCompleteProps } from 'antd';

const mockVal = (str: string, repeat = 1) => ({
  value: str.repeat(repeat),
});

const App: React.FC = () => {
  const [options, setOptions] = useState<AutoCompleteProps['options']>([]);
  const [anotherOptions, setAnotherOptions] =
useState<AutoCompleteProps['options']>([]);

  const getPanelValue = (searchText: string) =>
    !searchText ? [] : [mockVal(searchText), mockVal(searchText, 2),
mockVal(searchText, 3)];

  return (
    <Space direction="vertical" style={{ width: '100%' }}>
      <AutoComplete
        options={options}
        onSearch={(text) => setOptions(getPanelValue(text))}
        status="error"
        style={{ width: 200 }}
      />
      <AutoComplete
        options={anotherOptions}
        onSearch={(text) => setAnotherOptions(getPanelValue(text))}
        status="warning"
        style={{ width: 200 }}
      />
    </Space>
  );
};

export default App;
```

多种形态

v5.13.0

```
import React, { useState } from 'react';
import { AutoComplete, Flex } from 'antd';
import type { AutoCompleteProps } from 'antd';

const mockVal = (str: string, repeat = 1) => ({
  value: str.repeat(repeat),
});

const App: React.FC = () => {
  const [options, setOptions] = useState<AutoCompleteProps['options']>([]);

  const getPanelValue = (searchText: string) =>
    !searchText ? [] : [mockVal(searchText), mockVal(searchText, 2),
mockVal(searchText, 3)];

  return (
    <Flex vertical gap={12}>
      <AutoComplete
        options={options}
        style={{ width: 200 }}
        placeholder="Outlined"
        onSearch={(text) => setOptions(getPanelValue(text))}
        onSelect={globalThis.console.log}
      />
      <AutoComplete
        options={options}
        style={{ width: 200 }}
        placeholder="Filled"
        onSearch={(text) => setOptions(getPanelValue(text))}
        onSelect={globalThis.console.log}
        variant="filled"
      />
      <AutoComplete
        options={options}
        style={{ width: 200 }}
        placeholder="Borderless"
        onSearch={(text) => setOptions(getPanelValue(text))}
        onSelect={globalThis.console.log}
        variant="borderless"
      />
    </Flex>
  );
};

export default App;
```

自定义清除按钮

```
import React, { useState } from 'react';
import { CloseSquareFilled } from '@ant-design/icons';
import { AutoComplete } from 'antd';
import type { AutoCompleteProps } from 'antd';

const mockVal = (str: string, repeat = 1) => ({
  value: str.repeat(repeat),
});

const App: React.FC = () => {
  const [options, setOptions] = useState<AutoCompleteProps['options']>([]);

  const getPanelValue = (searchText: string) =>
    !searchText ? [] : [mockVal(searchText), mockVal(searchText, 2),
mockVal(searchText, 3)];

  return (
    <>
      <AutoComplete
        options={options}
        style={{ width: 200 }}
        onSearch={(text) => setOptions(getPanelValue(text))}
        placeholder="UnClearable"
        allowClear={false}
      />
      <br />
      <br />
      <AutoComplete
        options={options}
        style={{ width: 200 }}
        onSearch={(text) => setOptions(getPanelValue(text))}
        placeholder="Customized clear icon"
        allowClear={{ clearIcon: <CloseSquareFilled /> }}
      />
    </>
  );
};

export default App;
```

在 Form 中 Debug

Debug

```
import React from 'react';
import { SearchOutlined } from '@ant-design/icons';
import { AutoComplete, Button, Form, Input, TreeSelect } from 'antd';

const formItemLayout = {
  labelCol: {
    xs: { span: 24 },
    sm: { span: 8 },
  },
  wrapperCol: {
    xs: { span: 24 },
    sm: { span: 16 },
  },
};

const App: React.FC = () => (
  <Form style={{ margin: '0 auto' }} {...formItemLayout}>
    <Form.Item label="单独 AutoComplete">
      <AutoComplete />
    </Form.Item>
    <Form.Item label="单独 TreeSelect">
      <TreeSelect />
    </Form.Item>
    <Form.Item label="添加 Input.Group 正常">
      <Input.Group compact>
        <TreeSelect style={{ width: '30%' }} />
        <AutoComplete />
      </Input.Group>
    </Form.Item>
    <Form.Item label="包含 search 图标正常">
      <AutoComplete>
        <Input suffix={<SearchOutlined />} />
      </AutoComplete>
    </Form.Item>
    <Form.Item label="同时有 Input.Group 和图标发生移位">
      <Input.Group compact>
        <TreeSelect style={{ width: '30%' }} />
        <AutoComplete>
          <Input suffix={<SearchOutlined />} />
        </AutoComplete>
      </Input.Group>
    </Form.Item>
    <Form.Item label="同时有 Input.Group 和 Search 组件发生移位">
      <Input.Group compact>
        <TreeSelect style={{ width: '30%' }} />
        <AutoComplete>
          <Input.Search />
        </AutoComplete>
      </Input.Group>
    </Form.Item>
    <Form.Item label="Input Group 和 Button 结合">
      <Input.Group compact>
        <TreeSelect style={{ width: '20%' }} />
        <AutoComplete>
          <Input.Search />
        </AutoComplete>
        <Button type="primary" icon={<SearchOutlined />}>
          Search
        </Button>
      </Input.Group>
    </Form.Item>
  </Form>
);

export default App;
```

AutoComplete 和 Select

Debug

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

const AutoCompleteAndSelect = () => {
  return (
    <Flex vertical gap={16}>
      {(['small', 'middle', 'large'] as const).map((size) => (
        <Flex key={size}>
```

```
          <Select
            value="centered"
            size={size}
            style={{ width: 200 }}
            searchValue="centered"
            showSearch
          />
          <AutoComplete value="centered" size={size} style={{ width: 200 }}
/>
        </Flex>
      ))}
    </Flex>
  );
};

export default AutoCompleteAndSelect;
```

_InternalPanelDoNotUseOrYouWillBeFired

Debug

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

const { _InternalPanelDoNotUseOrYouWillBeFired: InternalAutoComplete } =
AutoComplete;

const App: React.FC = () => {
  const [open, setOpen] = React.useState(false);

  return (
    <Space direction="vertical" style={{ display: 'flex' }}>
      <Switch checked={open} onChange={() => setOpen(!open)} />
      <InternalAutoComplete
        defaultValue="lucy"
        style={{ width: 120 }}
        open={open}
        options={[
          { label: 'Jack', value: 'jack' },
          { label: 'Lucy', value: 'lucy' },
          { label: 'Disabled', value: 'disabled' },
          { label: 'Bamboo', value: 'bamboo' },
        ]}
      />
    </Space>
  );
};

export default App;
```

API

通用属性参考：通用属性

参数	说明	类型	
allowClear	支持清除	boolean | { clearIcon?: ReactNode }	fal
autoFocus	自动获取焦点	boolean	fal
backfill	使用键盘选择选项的时候把选中项回填到输入框中	boolean	fal
children (自动完成的数据源)	自动完成的数据源，不能和自定义输入框同时配置	React.ReactElement<OptionProps> | Array<React.ReactElement<OptionProps>>	-
children (自定义输入框)	自定义输入框，不能和自动完成的数据源同时配置	HTMLInputElement | HTMLTextAreaElement | React.ReactElement<InputProps>	<Ir
defaultActiveFirstOption	是否默认高亮第一个选项	boolean	tru
defaultOpen	是否默认展开下拉菜单	boolean	-
defaultValue	指定默认选中的条目	string	-
disabled	是否禁用	boolean	fal
dropdownRender	自定义下拉框内容	(menus: ReactNode) => ReactNode	-
popupClassName	下拉菜单的 className 属性	string	-
popupMatchSelectWidth	下拉菜单和选择器同宽。默	boolean | number	tru

	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		
filterOption	是否根据输入项进行筛选。当其为一个函数时，会接收 `inputValue` `option` 两个参数，当 `option` 符合筛选条件时，应返回 true，反之则返回 false	boolean | function(inputValue, option)	tru
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)	() = do
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	-
options	数据化配置选项内容，相比 jsx 定义会获得更好的渲染性能	{ label, value }[]	-
placeholder	输入框提示	string	-
status	设置校验状态	'error' | 'warning'	-

size	控件大小	`large` | `middle` | `small`	-
value	指定当前选中的条目	string	-
variant	形态变体	`outlined` | `borderless` | `filled`	ou
virtual	设置 false 时关闭虚拟滚动	boolean	tru
onBlur	失去焦点时的回调	function()	-
onChange	选中 option，或 input 的 value 变化时，调用此函数	function(value)	-
onDropdownVisibleChange	展开下拉菜单的回调	function(open)	-
onFocus	获得焦点时的回调	function()	-
onSearch	搜索补全项的时候调用	function(value)	-
onSelect	被选中时调用，参数为选中项的 value 值	function(value, option)	-
onClear	清除内容时的回调	function	-
onInputKeyDown	按键按下时回调	(event: KeyboardEvent) => void	-
onPopupScroll	下拉列表滚动时的回调	(event: UIEvent) => void	-

方法

名称	描述	版本
blur()	移除焦点	
focus()	获取焦点	

主题变量（Design Token）

FAQ

为何受控状态下使用 onSearch 无法输入中文?

请使用 `onChange` 进行受控管理。`onSearch` 触发于搜索输入，与 `onChange` 时机不同。此外，点击选项时也不会触发 `onSearch` 事件。

相关 issue：[#18230](#18230) [#17916](#17916)

为何 options 为空时，受控 open 展开不会显示下拉菜单?

AutoComplete 组件本质上是 Input 输入框的一种扩展，当 `options` 为空时，显示空文本会让用户误以为该组件不可操作，实际上它仍然可以进行文本输入操作。因此，为了避免给用户带来困惑，当 `options` 为空时， `open` 属性为 `true` 也不会展示下拉菜单，需要与 `options` 属性配合使用。