Select 选择属性

选择器组件能从一个选项列表中去获得用户所提供的信息。

{{"component": "modules/components/ComponentLinkHeader.js"}}

基础的选择器

我们通常将菜单（Menus）放置在其所点击的元素上，这样的话能够确保当前选定的菜单项显示在点击的元素之上。

{{"demo": "BasicSelect.js"}}

高级功能

Select 组件的设计原理是和一个原生的 `<select>` 元素能够互相替代。

若您需要一个更优雅的功能，譬如 combobox，multiselect，autocomplete，async 或者 creatable support，请查看 `Autocomplete` 组件。 此组件旨在改进 “react-select” 和 “downshift” 这两个包。

属性

选择器组件是通过自定义 InputBase 的 `<input>` 元素来实现的。 It extends the text field components sub-components, either the OutlinedInput, Input, or FilledInput, depending on the variant selected. 它有着相同的样式和许多相同的属性。 It shares the same styles and many of the same props. 详情请参阅相应组件的 API 文档。

Filled and standard variants

{{"demo": "SelectVariants.js"}}

标签和助手文本

{{"demo": "SelectLabels.js"}}

> *⚠ Note that when using FormControl with the outlined variant of the Select, you need to provide a label in two places: in the InputLabel component and in the `label` prop of the Select component (see the above demo).*

自动宽度

{{"demo": "SelectAutoWidth.js"}}

其他属性

{{"demo": "SelectOtherProps.js"}}

原生选择器

为了提高用户体验，对于在移动设备上使用平台的原生选择器这样的模式，我们是支持的。

{{"demo": "NativeSelect.js"}}

TextField

`TextField` wrapper 组件是一个完整的表单控件，它包括了标签，输入和帮助文本。 您可以在 在此章节中 查看使用 select 模式的示例。

自定义选择器

你可以参考以下一些例子来自定义组件。 您可以在 重写文档页面 中了解更多有关此内容的信息。

首先，需要设置 `InputBase` 组件的样式。 一旦设置好了样式，您就可以直接使用文本框组件，也可以将其作为一个 `select` 的字段提供给 select 组件的 `input` 属性。 Notice that the `"standard"` variant is easier to customize, since it does not wrap the contents in a `fieldset` / `legend` markup.

{{"demo": "CustomizedSelects.js"}}

🧐 If you are looking for inspiration, you can check MUI Treasury's customization examples.

多重选择

The `Select` component can handle multiple selections. It's enabled with the `multiple` prop.

与单项选择一样，您可以通过访问 `onChange` 的回调函数中的 `event.target.value` 来提取新的值。 它总是以一个数组的形式出现。

默认值

{{"demo": "MultipleSelect.js"}}

选中标记

{{"demo": "MultipleSelectCheckmarks.js"}}

Chip

{{"demo": "MultipleSelectChip.js"}}

占位符

{{"demo": "MultipleSelectPlaceholder.js"}}

原生（Native）

{{"demo": "MultipleSelectNative.js"}}

可被控制的打开选择框

You can control the open state of the select with the `open` prop. Alternatively, it is also possible to set the initial (uncontrolled) open state of the component with the `defaultOpen` prop.

{{"demo": "ControlledOpenSelect.js"}}

与对话框组件（Dialog）一起使用

While it's discouraged by the Material Design guidelines, you can use a select inside a dialog.

{{"demo": "DialogSelect.js"}}

联动

可以和 `ListSubheader` 组件一起罗列分类，或者和原生的 `<optgroup>` 元素一起使用。

{{"demo": "GroupedSelect.js"}}

无障碍设计

若想正确的给 `Select` 加上标签，你的 input 控件需要一个额外的带有 label 的 `id` 属性。 `id` 的内容需要和 `Select` 的 `labelId` 值相同，例如：

```
<InputLabel id="label">年龄</InputLabel>
<Select labelId="label" id="select" value="20">
  <MenuItem value="10">10</MenuItem>
  <MenuItem value="20">20</MenuItem>
</Select>
```

对于一个 原生选择器，你应该通过将选择元素的 `id` 属性的值赋给 `InputLabel` 的 `htmlFor` 属性来提及标签。

```
<TextField id="select" label="Age" value="20" select>
  <MenuItem value="10">Ten</MenuItem>
  <MenuItem value="20">Twenty</MenuItem>
</TextField>
```

或者，您也可以使用一个带有 `id` 和 `label` 的 `TextField` 组件来创建合适的标记和 id：

```
<InputLabel htmlFor="select">Age</InputLabel>
<NativeSelect id="select">
  <option value="10">Ten</option>
  <option value="20">Twenty</option>
</NativeSelect>
```

Unstyled

The Select also comes with an unstyled version. It's ideal for doing heavy customizations and minimizing bundle size.

Unstyled component

```
import SelectUnstyled from '@mui/base/SelectUnstyled';
```

Basic usage

{{"demo": "UnstyledSelectSimple.js"}}

The `SelectUnstyled` is a component that accepts generic props. Due to Typescript limitations, this may cause unexpected behavior when wrapping the component in `forwardRef` (or other higher-order components). In such

cases, the generic argument will be defaulted to `unknown` and type suggestions will be incomplete. To avoid this, manually cast the resulting component to the correct type (as shown above).

The rest of the demos below will not use `forwardRef` for brevity.

Controlled select

The SelectUnstyled can be used as either uncontrolled (as shown in the demo above) or controlled component.

{{"demo": "UnstyledSelectControlled.js"}}

Usage with object values

The unstyled select may be used with non-string values.

{{"demo": "UnstyledSelectObjectValues.js"}}

Customizing the selected value appearance

It is possible to customize the selected value display by providing a function to the `renderValue` prop. The element returned by this function will be rendered inside the select's button.

{{"demo": "UnstyledSelectCustomRenderValue.js"}}

Customizing the options' appearance

Options don't have to be plain strings. You can include custom elements to be rendered inside the listbox.

{{"demo": "UnstyledSelectRichOptions.js"}}

Grouping

Options can be grouped, similarly to the how the native `select` element works. Unlike the native `select`, however, the groups can be nested.

Place the `Option` components inside `OptionGroup` to achieve this.

{{"demo": "UnstyledSelectGrouping.js"}}

Multiselect

To be able to select multiple options at once, use the `MultiSelectUnstyled` component.

```
import { MultiSelectUnstyled } from '@mui/base/SelectUnstyled';
```

{{"demo": "UnstyledSelectMultiple.js"}}

useSelect hook

```
import { useSelect } from '@mui/base/SelectUnstyled';
```

If you need to use Select's functionality in another component, you can use the `useSelect` hook. It enables maximal customizability at the cost of being low-level.

The following example shows a select that opens when hovered over or focused. It can be controlled by a mouse/touch or a keyboard.

{{"demo": "UseSelect.js"}}