Seleção

Os componentes de seleção são usados para coletar informações fornecidas pelo usuário em uma lista de opções.

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

Seleção Simples

Os menus são posicionados sobre seus elementos emissores, de modo que o item de menu atualmente selecionado apareça na parte superior do elemento emissor.

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

Recursos avançados

O componente `Select` é pensado para ser intercambiável com um elemento nativo `<select>`.

Se você estiver procurando por recursos mais avançados, como combobox, seleção múltipla, autocompletar, uso assíncrono ou com suporte de adição, vá para o componente `Autocomplete`. A ideia dessa ferramenta é ser uma versão melhorada das bibliotecas "react-select" e "downshift".

Propriedades

O componente seleção é implementado como um elemento `<input>` personalizado do InputBase. It extends the text field components sub-components, either the OutlinedInput, Input, or FilledInput, depending on the variant selected. Ele compartilha os mesmos estilos e muitas das mesmas propriedades. Consulte a página da API do respectivo componente para obter detalhes.

Filled and standard variants

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

Rótulos e texto de ajuda

{{"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).*

Largura automática

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

Outras propriedades

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

Campos de Texto

Como a experiência do usuário pode ser melhorada em dispositivos móveis usando a seleção nativa da plataforma, permitimos esse padrão.

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

TextField

O componente wrapper `TextField` é um controle de formulário completo, incluindo um rótulo, entrada e texto de ajuda. Você pode encontrar um exemplo de seleção nesta seção.

Seleções customizadas

Aqui estão alguns exemplos de customização do componente. Você pode aprender mais sobre isso na página de documentação de sobrescritas.

O primeiro passo é estilizar o componente `InputBase`. Uma vez estilizado, você pode usá-lo diretamente como um campo de texto ou fornecê-lo à propriedade `input` da seleção para ter um campo `select`. Notice that the `"standard"` variant is easier to customize, since it does not wrap the contents in a `fieldset` / `legend` markup.

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

🤓 Se você está procurando inspiração, você pode verificar os exemplos de customização de MUI Treasury.

Seleção Aberta Controlada

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

Como na seleção única, você pode extrair o novo valor acessando `event.target.value` na chamada `onChange`. É sempre uma matriz.

Padrão

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

Marcações

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

Chip

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

Placeholder

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

Nativo

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

Seleção aberta controlada

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"}}

Com um diálogo

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

Exiba categorias com o componente `ListSubheader` ou com o elemento nativo `<optgroup>`.

Agrupando

Display categories with the `ListSubheader` component or the native `<optgroup>` element.

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

Acessibilidade

Para rotular corretamente seu campo `Select` você precisa de um elemento extra com um `id` que contenha o rótulo desejado. Esse `id` precisa coincidir com o `labelId` do `Select`, por exemplo.

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

Para uma seleção nativa, você deve utilizar um rótulo fornecendo o atributo `id` do elemento de seleção para o atributo `htmlFor` do `InputLabel`:

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

Alternativamente, um `TextField` com `id` e `label` cria a marcação adequada e ids para você:

```
<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"}}