Select

Select components are used for collecting user provided information from a list of options.

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

Basic select

Menus are positioned under their emitting elements, unless they are close to the bottom of the viewport.

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

Advanced features

The Select component is meant to be interchangeable with a native `<select>` element.

If you are looking for more advanced features, like combobox, multiselect, autocomplete, async or creatable support, head to the `Autocomplete` component. It's meant to be an improved version of the "react-select" and "downshift" packages.

Props

The Select component is implemented as a custom `<input>` element of the InputBase. It extends the text field components sub-components, either the OutlinedInput, Input, or FilledInput, depending on the variant selected. It shares the same styles and many of the same props. Refer to the respective component's API page for details.

Filled and standard variants

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

Labels and helper text

{{"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).*

Auto width

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

Small Size

{{"demo": "SelectSmall.js"}}

Other props

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

Native select

As the user experience can be improved on mobile using the native select of the platform, we allow such pattern.

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

TextField

The `TextField` wrapper component is a complete form control including a label, input and help text. You can find an example with the select mode in this section.

Customization

Here are some examples of customizing the component. You can learn more about this in the overrides documentation page.

The first step is to style the `InputBase` component. Once it's styled, you can either use it directly as a text field or provide it to the select `input` prop to have a `select` field. 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.

Multiple select

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

Like with the single selection, you can pull out the new value by accessing `event.target.value` in the `onChange` callback. It's always an array.

Default

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

Checkmarks

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

Chip

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

Placeholder

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

Native

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

Controlling the open state

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"}}

With a dialog

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

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

Grouping

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

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

Accessibility

To properly label your `Select` input you need an extra element with an `id` that contains a label. That `id` needs to match the `labelId` of the `Select` e.g.

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

Alternatively a `TextField` with an `id` and `label` creates the proper markup and ids for you:

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

For a native select, you should mention a label by giving the value of the `id` attribute of the select element to the `InputLabel` 's `htmlFor` attribute:

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