TextField 文本框

用户可以在文本框内输入或编辑文字。

用户可以通过文本框在界面中输入文本。 通常，我们会在表单域和对话框中使用它们。

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

基础文本框

`TextField` 包装器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。 It comes with three variants: outlined (default), filled, and standard.

{{"demo": "BasicTextFields.js"}}

友情提示： Material 设计指南不再记录 `TextField` 的 standard 布局，（原因见此），但是 Material-UI 会继续支持此布局。

Form props 表单的属性

该组件支持标准的表单属性，例如 `required`、`disabled`、`type` 以及 `helperText` 等，用于提供字段输入的上下文，例如输入的使用方式。

{{"demo": "FormPropsTextFields.js"}}

校验

The `error` prop toggles the error state. The `helperText` prop can then be used to provide feedback to the user about the error.

{{"demo": "ValidationTextFields.js"}}

多行属性

`multiline` 属性将文本框转换为 `<textarea>` 元素。 除非设置了 `rows` 属性，否则文本框的高度会动态匹配其内容（当你使用 TextareaAutosize 属性时）。 You can use the `minRows` and `maxRows` props to bound it.

{{"demo": "MultilineTextFields.js"}}

Select 选择属性

使用 `select` 属性的时候，您可以在文本框内插入一个 Select 组件。

{{"demo": "SelectTextFields.js"}}

Icons 图标

我们提供了在文本框内展示图标的不同的方式。

{{"demo": "InputWithIcon.js"}}

修饰输入框

一个主流的方法是使用 `InputAdornment` 组件。 This can be used to add a prefix, a suffix, or an action to an input. 例如，可以用一个图标按钮来隐藏或者显示输入框里的密码。 例如，可以用一个图标按钮来隐藏或者显示输入框里的密码。

{{"demo": "InputAdornments.js"}}

尺寸

想要使用外观看起来比较小的输入框吗？ 试着使用 `size` 属性吧。

{{"demo": "TextFieldSizes.js"}}

`filled` 变体的输入高度可以通过在外部渲染标签来降低。

{{"demo": "TextFieldHiddenLabel.js"}}

Margin

The `margin` prop can be used to alter the vertical spacing of the text field. Using `none` (default) doesn't apply margins to the `FormControl` whereas `dense` and `normal` do.

{{"demo": "LayoutTextFields.js"}}

全宽

我们提供了 `fullWidth` 属性，使用它的时候，输入框会占据整个容器的宽度。

{{"demo": "FullWidthTextField.js"}}

Uncontrolled vs. Controlled

你可以选择控制的组件，或者不受控的组件。

{{"demo": "StateTextFields.js"}}

Components 组件

`TextField` 是由一些较小的组件组成的（ `FormControl` ， `Input` ， `FilledInput` ， `InputLabel` ， `OutlinedInput` ， and `FormHelperText` ），而你可以直接使用这些小的组件，来自定制你的表单域输入框。

您可能注意到了， 和原生的 HTML input 组件相比， `TextField` 组件缺失了一些属性。 这是故意为之的。 The component takes care of the most used properties. Then, it's up to the user to use the underlying component shown in the following demo. Still, you can use `inputProps` (and `InputProps` , `InputLabelProps` properties) if you want to avoid some boilerplate.

{{"demo": "ComposedTextField.js"}}

Inputs 输入框

{{"demo": "Inputs.js"}}

Color 颜色

当使用 `color` 属性时，聚焦文本框时的高亮颜色会被改变。

{{"demo": "ColorTextFields.js"}}

自定义输入

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。

{{"demo": "CustomizedInputs.js"}}

Customization does not stop at CSS. You can use composition to build custom components and give your app a unique feel. Below is an example using the `InputBase` component, inspired by Google Maps.

{{"demo": "CustomizedInputBase.js", "bg": true}}

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

`useFormControl`

For advanced customization use cases, a `useFormControl()` hook is exposed. This hook returns the context value of the parent `FormControl` component.

API

```
import { useFormControl } from '@mui/material/FormControl';
```

返回结果

`value` (*object*):

- `value.adornedStart` (*bool*): Indicate whether the child `Input` or `Select` component has a start adornment.
- `value.setAdornedStart` (*func*): Setter function for `adornedStart` state value.
- `value.color` (*string*): The theme color is being used, inherited from `FormControl` `color` prop .
- `value.disabled` (*bool*): Indicate whether the component is being displayed in a disabled state, inherited from `FormControl` `disabled` prop.
- `value.error` (*bool*): Indicate whether the component is being displayed in an error state, inherited from `FormControl` `error` prop
- `value.filled` (*bool*): Indicate whether input is filled
- `value.focused` (*bool*): Indicate whether the component and its children are being displayed in a focused state
- `value.fullWidth` (*bool*): Indicate whether the component is taking up the full width of its container, inherited from `FormControl` `fullWidth` prop
- `value.hiddenLabel` (*bool*): Indicate whether the label is being hidden, inherited from `FormControl` `hiddenLabel` prop
- `value.required` (*bool*): Indicate whether the label is indicating that the input is required input, inherited from the `FormControl` `required` prop
- `value.size` (*string*): The size of the component, inherited from the `FormControl` `size` prop
- `value.variant` (*string*): The variant is being used by the `FormControl` component and its children, inherited from `FormControl` `variant` prop
- `value.onBlur` (*func*): Should be called when the input is blurred

- `value.onFocus` (*func*): Should be called when the input is focused
- `value.onEmpty` (*func*): Should be called when the input is emptied
- `value.onFilled` (*func*): Should be called when the input is filled

示例

{{"demo": "UseFormControl.js"}}

设计局限

缩放

输入框标签的 "shrink" 状态并不总是正确的。 输入标签应在输入框显示内容的时候立即收缩。 在某些情况下，我们无法确定输入框的 "shrink" 状态 (如数字输入、日期时间输入、条带输入)。 这样一来，有可能出现重叠的现象。

 shrink 缩放

若要解决此问题，您可以在输入框的标签上强制赋予 "shrink" 状态。

```
<TextField InputLabelProps={{ shrink: true }} />
```

或

```
<InputLabel shrink>计数</InputLabel>
```

悬浮的标签

The floating label is absolutely positioned. It won't impact the layout of the page. Make sure that the input is larger than the label to display correctly.

type="number"

type="number" 的输入存在潜在的可用性问题。

- Allowing certain non-numeric characters ('e', '+', '-', '.') and silently discarding others
- 滚动增加/减少数字的功能可能会导致意外和难以察觉的变化。

有关该话题的更多信息 - 请参阅 GOV.UK 设计系统团队的 这篇文章，来了解更详细的解释。

对于数字验证，一个可行的替代方法是使用默认的 type="text" 和 *pattern* 属性，例如：

```
<TextField inputProps={{ inputMode: 'numeric', pattern: '[0-9]*' }} />
```

以后我们可能会提供 数字（number）输入组件。

辅助文本

助手文本属性会影响文本框的高度。 如果两个文本框并排放置，一个有辅助文本，另一个没有，那么它们的高度就会不同。 例如:

{{"demo": "HelperTextMisaligned.js"}}

This can be fixed by passing a space character to the `helperText` prop:

{{"demo": "HelperTextAligned.js"}}

与第三方 input 库的整合

您可以使用第三方库来格式化您的输入框。 只要确保在整合的时候，您提供了一个带有 `inputComponent` 属性的自定义 `<input>` 元素。

下面的演示使用 react-text-mask 和 react-number-format 这两个基本库。 同样的概念可以适用于 这个例子：react-stripe-element。

{{"demo": "FormattedInputs.js"}}

第三方所提供的输入组件应该暴露一个 ref，其值实现以下接口：

```
interface InputElement {
  focus(): void;
  value?: string;
}
```

```
const MyInputComponent = React.forwardRef((props, ref) => {
  const { component: Component, ...other } = props;

  // 实现 `InputElement` 接口
  React.useImperativeHandle(ref, () => ({
    focus: () => {
      // 在这里提供第三方组件的聚焦（focus）渲染方法
    },
    // 隐藏值，例如 react-stripe-elements
  }));

  // `Component` 将会是下面例子中的 `SomeThirdPartyComponent`
  return <Component {...other} />;
});

// 使用
<TextField
  InputProps={{
    inputComponent: MyInputComponent,
    inputProps: {
      component: SomeThirdPartyComponent,
    },
  }}
/>;
```

无障碍设计

为了确保您的文本框可提供无障碍访问，**输入框必须和标签以及帮助文本连在一起调用。** 而且，深层的 DOM 节点应该遵循这个结构。

```
<div class="form-control">
  <label for="my-input">电子邮件</label>
  <input id="my-input" aria-describedby="my-helper-text" />
  <span id="my-helper-text">我们绝不会分享您的邮件地址。 </span>
</div>
```

- If you are using the `TextField` component, you just have to provide a unique `id` unless you're using the `TextField` only client side. Until the UI is hydrated `TextField` without an explicit `id` will not have associated labels.
- If you are composing the component:

```
<FormControl>
  <InputLabel htmlFor="my-input">电子邮件</InputLabel>
  <Input id="my-input" aria-describedby="my-helper-text" />
  <FormHelperText id="my-helper-text">我们绝不会分享您的邮件地址。 </FormHelperText>
</FormControl>
```

补充项目

For more advanced use cases, you might be able to take advantage of:

- react-hook-form：用于表单验证的 React 钩子。
- formik-material-ui: Bindings for using MUI with formik.
- redux-form-material-ui：用于 Redux Form 和 Material-UI 捆绑使用。
- mui-rff：用于 React Final Form 和 Material-UI 捆绑使用。

Unstyled

For advanced customization scenarios, you can use the unstyled primitives.

The basic building blocks are the `InputUnstyled` component and the `useInput` hook.

Unstyled component

The `InputUnstyled` component wraps the native `input` or `textarea` element. You can, optionally, provide a custom component to be rendered instead.

{{"demo": "UnstyledInput.js"}}

Hook

The `useInput` hook is the headless version of the `InputUnstyled` component. Use it for even greater control over the rendered output.

{{"demo": "UseInput.js"}}