Autocompletar

O autocompletar é uma entrada de texto normal aprimorada por um painel de opções sugeridas.

Essa ferramenta é útil para configurar os valores de um campo de texto quando em um dos dois cenários abaixo:

1. O valor para a caixa de texto deve ser escolhido a partir de um conjunto pré-definido de valores permitidos, por exemplo, um campo de localização deve conter um nome de localização válido: caixa de combinação.
2. A caixa de texto pode conter qualquer valor arbitrário, mas é mais vantajosa, porque pode sugerir possíveis valores para o usuário, por exemplo, um campo de pesquisa que pode sugerir pesquisas anteriores ou semelhantes para economizar o tempo do usuário: free solo.

A ideia dessa ferramenta é ser uma versão melhorada das bibliotecas "react-select" e "downshift".

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

Caixa de combinação

O valor deve ser escolhido a partir de um conjunto predefinido de valores permitidos.

{{"demo": "ComboBox.js"}}

Estrutura de opções

Por padrão, o componente aceita as seguintes estruturas de opções:

```
interface AutocompleteOption {
  label: string;
}
// ou
type AutocompleteOption = string;
```

por exemplo:

```
const options = [
  { label: 'The Godfather', id: 1 },
  { label: 'Pulp Fiction', id: 2 },
];
// ou
const options = ['The Godfather', 'Pulp Fiction'];
```

No entanto, você pode usar estruturas diferentes fornecendo a propriedade `getOptionLabel`.

Área de exemplos

Cada um dos exemplos a seguir demonstra uma funcionalidade do componente Autocomplete.

{{"demo": "Playground.js"}}

Seleção de países

Escolha um dos 248 países.

{{"demo": "CountrySelect.js"}}

Estados controlados

O componente tem dois estados que podem ser controlados:

1. o estado "value" com a combinação das propriedades `value` / `onChange`. Esse estado representa o valor selecionado pelo usuário, por exemplo, quando pressionando `Enter`.
2. o estado "input value" com a combinação das propriedades `inputValue` / `onInputChange`. Esse estado representa o valor exibido na caixa de texto.

> ⚠️ *Esses dois estados são isolados e devem ser controlados de forma independente.*

{{"demo": "ControllableStates.js"}}

Free solo

Coloque `freeSolo` como true para que o campo de texto contenha qualquer valor aleatório.

Campo search

A propriedade é projetada para cobrir o principal caso de uso de uma **caixa de pesquisa** com sugestões, por exemplo, pesquisa do Google ou react-autowhatever.

{{"demo": "FreeSolo.js"}}

Creatable

Se você pretende usar este modo para uma caixa de combinação, por experiência (uma versão aprimorada de um elemento select) recomendamos a configuração:

- `selectOnFocus` para ajudar o usuário a limpar o valor selecionado.
- `clearOnBlur` para ajudar o usuário a digitar um novo valor.
- `handleHomeEndKeys` para mover o foco dentro do popup com as teclas `Home` e `End`.
- Adicione uma última opção para indicar a possibilidade de adição, por exemplo `Adicionar "SUA PESQUISA"`.

{{"demo": "FreeSoloCreateOption.js"}}

Você pode também exibir um diálogo quando o usuário quiser adicionar um novo valor.

{{"demo": "FreeSoloCreateOptionDialog.js"}}

Agrupamento

Você pode agrupar as opções com a propriedade `groupBy`. Se você fizer isso, certifique-se de que as opções também estejam classificadas com a mesma dimensão que serão agrupadas, caso contrário, você notará cabeçalhos duplicados.

{{"demo": "Grouped.js"}}

Opções desabilitadas

{{"demo": "DisabledOptions.js"}}

`useAutocomplete`

For advanced customization use cases, a headless `useAutocomplete()` hook is exposed. It accepts almost the same options as the Autocomplete component minus all the props related to the rendering of JSX. O componente de auto completar é baseado neste hook.

```
import { useAutocomplete } from '@mui/base/AutocompleteUnstyled';
```

O hook `useAutocomplete` também é reexportado de @mui/material por conveniência e compatibilidade com versões anteriores.

```
import { createFilterOptions } from '@material-ui/core/Autocomplete';
```

- 4.5 kB gzipado.

{{"demo": "UseAutocomplete.js", "defaultCodeOpen": false}}

Hook customizado

{{"demo": "CustomizedHook.js"}}

Vá para a seção customização para um exemplo com o componente `Autocomplete` em vez do hook.

Requisições assíncronas

O componente suporta dois cenários de uso assíncrono diferentes:

- Carregar ao abrir: espera uma interação com o componente para carregar as opções.
- Pesquisar enquanto digita: um novo pedido é feito para cada tecla pressionada.

Carregar ao abrir

Exibe um estado de progresso enquanto a requisição de rede estiver pendente.

{{"demo": "Asynchronous.js"}}

Pesquisar enquanto digita

Se sua lógica está buscando novas opções em cada tecla pressionada e usando o valor atual da caixa de texto para filtrar no servidor, você pode querer considerar um limite nas requisições.

Uma customização de UI para o autocompletar de lugares do Google Maps.

```
<Autocomplete filterOptions={(x) => x} />
```

Lugares com a API do Google Maps

Uma customização de UI para o autocompletar de lugares do Google Maps. For this demo, we need to load the Google Maps JavaScript and Google Places API.

{{"demo": "GoogleMaps.js"}}

> ⚠️ *Antes de você começar a usar a API JavaScript do Google Maps você precisará estar cadastrado e ter uma conta.*

Múltiplos valores

Also known as tags, the user is allowed to enter more than one value.

{{"demo": "Tags.js"}}

Opções fixas

In the event that you need to lock certain tags so that they can't be removed, you can set the chips disabled.

{{"demo": "FixedTags.js"}}

Caixas de seleção

{{"demo": "CheckboxesTags.js"}}

Limitar tags

You can use the `limitTags` prop to limit the number of displayed options when not focused.

Você pode usar a propriedade `limitTags` para limitrar o número de opções exibidas quando o componente não estiver com o foco.

Tamanhos

Gosta mais de campos de texto menores? Use a propriedade `size`.

{{"demo": "Sizes.js"}}

Customização

Input customizado

A propriedade `renderInput` permite que você customize o input renderizado. O primeiro argumento desta propriedade de render, contém propriedades que você precisa encaminhar. Preste atenção específicamente nas chaves `ref` e `inputProps`.

{{"demo": "CustomInputAutocomplete.js"}}

Seletor do GitHub

This demo reproduces GitHub's label picker:

{{"demo": "GitHubLabel.js"}}

Head to the Customized hook section for a customization example with the `useAutocomplete` hook instead of the component.

Realce

Va para a seção Hook customizado para um exemplo com o uso do hook customizado `useAutocomplete` ao invés do componente.

{{"demo": "Highlights.js"}}

Filtro customizado

O componente expõe uma fábrica para criar um método de filtro que pode ser fornecido para a propriedade `filterOptions`. Você pode usar ela para modificar o comportamento padrão do filtro.

```
import { createFilterOptions } from '@mui/material/Autocomplete';
```

`createFilterOptions(config) => filterOptions`

Argumentos

1. `config` (*object* [opcional]):

- `config.ignoreAccents` (*bool* [optional]): Padrão como `verdadeiro`. Remover sinais diacríticos.
- `config.ignoreCase` (*boolean* [optional]): Padrão como `verdadeiro`. Minúsculas em tudo.
- `config.limit` (*number* [opcional]): Padrão null. Limitar o número de opções sugeridas a serem exibidas. Por exemplo, se `config.limit` é `100`, somente as primeiras `100` opções correspondentes são exibidas. Isto pode ser útil se um monte corresponderem e a virtualização não estiver configurada.
- `config.matchFrom` ('*any*' | '*start*' [opcional]): Padrão `'any'`.
- `config.stringify` (*func* [opcional]): Controla a forma como a opção é convertida em texto, dessa forma pode ser comparada com qualquer fragmento de texto.
- `config.trim` (*bool* [opcional]): Padrão `false`. Remover espaços ao fim.

Retornos

`filterOptions`: the returned filter method can be provided directly to the `filterOptions` prop of the `Autocomplete` component, or the parameter of the same name for the hook.

`filterOptions`: o método de filtro retornado pode ser fornecido diretamente para a propriedade `filterOptions` do componente `Autocomplete` ou para o parâmetro de mesmo nome no hook.

```
const filterOptions = createFilterOptions({
  matchFrom: 'start',
  stringify: (option) => option.title,
});

<Autocomplete filterOptions={filterOptions} />;
```

Na demonstração a seguir, as opções necessárias para o filtro ser aplicado no inicio das opções:

Avançado

Para mecanismos de filtragem mais ricos, como correspondência difusa, recomenda-se explorar o match-sorter. Por exemplo:

```
import { matchSorter } from 'match-sorter';

const filterOptions = (options, { inputValue }) => matchSorter(options, inputValue);

<Autocomplete filterOptions={filterOptions} />;
```

Virtualização

Pesquise dentro de 10.000 opções geradas aleatoriamente. A lista é virtualizada graças a react-window.

{{"demo": "Virtualize.js"}}

Eventos

Se você deseja evitar o comportamento padrão do teclado, você pode definir a propriedade do evento `defaultMuiPrevented` para `true`:

```
<Autocomplete
  onKeyDown={(event) => {
    if (event.key === 'Enter') {
      // Previne o comportamento padrão do 'Enter'.
      event.defaultMuiPrevented = true;
      // seu código manipulador
    }
  }}
/>
```

Limitações

autocomplete/autofill

Os navegadores têm heurística para ajudar os usuários a preencherem os campos do formulário. No entanto, isso pode prejudicar a experiência do usuário com o componente.

Por padrão, o componente desabilita a entrada**autocomplete**(lembra o que o usuário digitou para um determinado campo em uma sessão anterior) com o atributo `autoComplete="off"` Atualmente, o Google Chrome não suporta essa configuração de atributo (Issue 587466). Google Chrome does not currently support this attribute setting (Issue 587466). Uma solução alternativa possível é remover o `id` para que o componente gere um aleatório.

No entanto, além de relembrar valores fornecidos anteriormente, o navegador também pode propor sugestões de **autofill** (preenchimento automático para informações de login, endereço ou detalhes de pagamento). No caso de você querer evitar o recurso de preenchimento automático, tente o seguinte:

- Nomeie o campo sem fornecer informações para o navegador do que ele representa. `id="field1"` ao invés de `id="country"`. Se você deixar o id do vazio, o componente utiliza um id aleatório.
- Defina `autoComplete="new-password"` (alguns navegadores irão sugerir uma senha forte para entradas com esta configuração de atributo):

```
<TextField
  {...params}
  inputProps={{
    ...params.inputProps,
    autoComplete: 'new-password',
  }}
/>
```

Leia [este guia na MDN] para mais detalhes.

iOS VoiceOver

VoiceOver no Safari do iOS não suporta o atributo `aria-owns` muito bem. Você pode contornar o problema com a propriedade `disablePortal`.

ListboxComponent

Se você fornecer um componente customizado na propriedade `ListboxComponent`, você precisará certificar-se de que o contêiner de scroll esteja com o atributo `role` definido como `listbox`. Isto garante o comportamento correto do scroll, por exemplo, quando utilizar o teclado para navegar.

Acessibilidade

(WAI-ARIA: https://www.w3.org/TR/wai-aria-practices/#combobox)

Incentivamos a utilização de um rótulo para a caixa de texto. O componente implementa as práticas de autoria da WAI-ARIA.