Tabs 选项卡

使用选项卡，你可以轻松地浏览和切换不同的视图。

对于在同一层次，并且息息相关的内容组，使用选项卡能够将它们分组并且在其之间切换。

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

基础选项卡

A basic example with tab panels.

{{"demo": "BasicTabs.js"}}

实验性的 API

遵循 WAI-ARIA 项目实践， `@material-ui/lab` 提供了工具集组件，该组件通过注入属性的方式来实现无障碍设计的选项卡。

{{"demo": "LabTabs.js"}}

包装的标签

Long labels will automatically wrap on tabs. If the label is too long for the tab, it will overflow, and the text will not be visible.

{{"demo": "TabsWrappedLabel.js"}}

Colored tab

{{"demo": "ColorTabs.js"}}

禁用选项卡

A tab can be disabled by setting the `disabled` prop.

{{"demo": "DisabledTabs.js"}}

固定的选项卡

Fixed tabs should be used with a limited number of tabs, and when a consistent placement will aid muscle memory.

全宽

若是较小的视图，则应使用 `variant="fullWidth"` 属性。 在这个演示中你还可以借鉴用 react-swipeable-views 来设置选项卡的过渡动画，并且在使用触摸设备时滑动标签。

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

居中对齐

而对于较大的视图，则应使用 `centered` 属性。

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

可滚动的选项卡

自动显示滚动按钮

左右滚动按钮将自动在桌面显示，并在移动设备上隐藏。 （基于视图宽度）

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

强制显示滚动按钮

通过使用 `scrollButtons={true}` `allowScrollButtonsMobile` 属性，无论当前视图宽度如何，都会显示左右的滚动按钮（保留空间）

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

如果你想确保按钮始终可见，那么你应该自定义不透明度：

```css
.MuiTabs-scrollButtons.Mui-disabled {
  opacity: 0.3;
}
```

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

永久隐藏滚动按钮

你可以使用 `scrollButtons={false}` 属性来永远隐藏左右的滚动按钮。 All scrolling must be initiated through user agent scrolling mechanisms (e.g. left/right swipe, shift mouse wheel, etc.)

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

自定义的选项卡

以下是自定义组件的一个示例。 您可以在 [重写文档页面](#) 中了解更多有关此内容的信息。

{{"demo": "CustomizedTabs.js"}}

🎨 如果您还在寻找灵感，您可以看看 [MUI Treasury 特别定制的一些例子](#)。

垂直的选项卡

使用 `orientation="vertical"` 来使垂直标签代替默认的水平标签。

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

请注意，你可以使用 `visibleScrollbar` 来恢复显示滚动条。

Nav tabs

By default, tabs use a `button` element, but you can provide your custom tag or component. 下面是一个实现导航选项卡的例子： 下面是一个实现导航选项卡的例子： Here's an example of implementing tabbed navigation:

{{"demo": "NavTabs.js"}}

Icon tabs

选项卡的标签可以是所有的图标或者所有的文本。

{{"demo": "IconTabs.js"}}

{{"demo": "IconLabelTabs.js"}}

Third-party routing library（第三方路由库）

By default, the icon is positioned at the `top` of a tab. Other supported positions are `start` , `end` , `bottom` . Other supported positions are `start` , `end` , `bottom` .

{{"demo": "IconPositionTabs.js"}}

无障碍设计

One frequent use case is to perform navigation on the client only, without an HTTP round-trip to the server. The `Tab` component provides the `component` prop to handle this use case. Here is a more detailed guide.

Accessibility

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

您需要采取以下步骤，来为无障碍技术提供一些必要的信息：

1. 在 `Tabs` 上应用 `aria-label` 或 `aria-labelledby` 标签。
2. 通过设置 `id` 、 `aria-controls` 和 `aria-labelledby` ， `Tab` 需要连接到其对应的 `[role="tabpanel"]` 。

实现这样的设计例子可以在本页面的演示中找到。 我们还在 `@material-ui/lab` 中发布了不需要额外工作就能使用的 一个实验性的 API。

键盘导航

该组件使用"手动激活"的行为来实现键盘导航。 如果你想切换到"选择自动跟随焦点"（selection automatically follows focus）的行为，你必须将 `selectionFollowsFocus` 传递给 `Tabs` 组件。 WAI-ARIA 项目实践中有一个详细的指南关于 how to decide when to make selection automatically follow focus。

演示

下面的两个演示只是在键盘导航行为上有所区别。 Focus a tab and navigate with arrow keys to notice the difference, e.g. `Arrow Left`.

```
/* 那个跟随焦点的选项卡 */
<Tabs selectionFollowsFocus />
```

{{"demo": "AccessibleTabs1.js", "defaultCodeOpen": false}}

```
/* Tabs where each tab needs to be selected manually */
<Tabs />
```

{{"demo": "AccessibleTabs2.js", "defaultCodeOpen": false}}

Unstyled

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

Unstyled component

```
import TabsUnstyled from '@mui/base/TabsUnstyled';
import TabsListUnstyled from '@mui/base/TabUnstyled';
import TabUnstyled from '@mui/base/TabUnstyled';
import TabPanelUnstyled from '@mui/base/TabPanelUnstyled';
```

{{"demo": "UnstyledTabsBasic.js"}}

Customizing the root element

By default, the `TabUnstyled` renders a native `button` element. You are free to override this by setting the `component` or `components.Root` prop. If a non-interactive element (such as a span) is provided this way, the `TabUnstyled` will take care of adding accessibility attributes. You are free to override this by setting the `component` or `components.Root` prop. If a non-interactive element (such as a span) is provided this way, the `TabUnstyled` will take care of adding accessibility attributes.

The `TabPanelUnstyled` on the other hand renders a native `div` element by default. You are free to override this as well by setting the `component` or `components.Root` prop on the `TabPanelUnstyled`.

{{"demo": "UnstyledTabsCustomized.js"}}