Transições

Transições ajudam a deixar a interface expressiva e fácil de usar.

Material-UI provê um número de transições que podem ser usadas para introduzir alguns movimentos básicos para os componentes de sua aplicação.

A paleta com funções de estilo.

Collapse

Expandir para fora partindo do centro do elemento filho. Use a propriedade `orientation` se você precisar de um colapso na horizontal. A propriedade `collapsedHeight` pode ser usada para definir a altura mínima quando não expandida.

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

Fade

Fade in de transparente para opaco.

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

Grow

Expands outwards from the center of the child element, while also fading in from transparent to opaque.

Expande para fora a partir do centro do elemento filho, enquanto também desvanece em efeito de transparente para opaco.

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

Slide

Deslize a partir da borda da tela. A propriedade `direction` controla em qual borda da tela a transição começa.

A propriedade `mountOnEnter` do componente de transição impede que o componente filho seja montado até que `in` seja `true`. Isso evita que o componente relativamente posicionado role para a visão a partir da posição de fora da tela. Da mesma forma, a propriedade `unmountOnExit` remove o componente do DOM após a transição ter sido exibida para fora da tela.

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

Slide relative to a container

The Slide component also accepts `container` prop, which is a reference to a DOM node. If this prop is set, the Slide component will slide from the edge of that DOM node.

Expandir para fora partindo do centro do elemento filho.

Zoom

Expandir para fora partindo do centro do elemento filho.

Este exemplo também demonstra como atrasar a transição de entrada.

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

Child requirement

- Para um melhor suporte a renderização no servidor, Material-UI provê uma propriedade `style` para o elemento filho de alguns componentes de transição, (Fade, Grow, Zoom, Slide). A propriedade `style` deve ser aplicada ao DOM para que a animação funcione conforme esperada.
- **Forward the ref**: The transition components require the first child element to forward its ref to the DOM node. For more details about ref, check out Caveat with refs
- **Single element**: The transition components require only one child element (`React. Fragment` is not allowed).

```
// O objeto `props` contém uma propriedade `style`.
// Você precisa fornecê-lo ao elemento `div` como mostrado aqui.
function MyComponent(props) {
  return (
    <div {...props}>
      Fade
    </div>
  );
}

export default Main() {
  return (
    <Fade>
      <MyComponent />
    </Fade>
  );
}
```

TransitionGroup

To animate a component when it is mounted or unmounted, you can use the `TransitionGroup` component from *react-transition-group*. As components are added or removed, the `in` prop is toggled automatically by `TransitionGroup` .

{{"demo": "TransitionGroupExample.js"}}

Propriedade TransitionComponent

Alguns componentes do Material-UI usam essas transições internamente. Estas aceitam uma propriedade `TransitionComponent` para customizar a transição padrão. Você pode usar qualquer um dos componentes acima ou seu próprio componente. Ele deve respeitar as seguintes condições:

- Aceitar uma propriedade `in` . Isso corresponde ao estado de aberto/fechado.
- Chamar a propriedade de callback `onEnter` quando a transição de entrada iniciar.
- Chamar a propriedade de callback `onExited` quando a transição de saída for concluída. Esses dois callbacks permitem desmontar os elementos filhos quando em estado fechado e totalmente transitados.

For more information on creating a custom transition, visit the *react-transition-group* `Transition` documentation.
Você também pode visitar as seções dedicadas de alguns dos componentes:

- Modal
- Dialog
- Popper
- Snackbar
- Tooltip

Performance & SEO

The content of transition component is mounted by default even if `in={false}`. This default behavior has server-side rendering and SEO in mind. If you render expensive component trees inside your transition it might be a good idea to change this default behavior by enabling the `unmountOnExit` prop:

```
<Fade in={false} unmountOnExit />
```

As with any performance optimization this is not a silver bullet. Be sure to identify bottlenecks first and then try out these optimization strategies.