React Transition component

Transitions

Transitions help to make a UI expressive and easy to use.

MUI provides transitions that can be used to introduce some basic motion to your applications.

{{“component”: “modules/components/ComponentLinkHeader.js”, “design”: false}}

Collapse

Expand from the start edge of the child element. Use the `orientation` prop if you need a horizontal collapse. The `collapsedSize` prop can be used to set the minimum width/height when not expanded.

{{“demo”: “SimpleCollapse.js”, “bg”: true}}

Fade

Fade in from transparent to opaque.

{{“demo”: “SimpleFade.js”, “bg”: true}}

Grow

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

The second example demonstrates how to change the `transform-origin`, and conditionally applies the `timeout` prop to change the entry speed.

{{“demo”: “SimpleGrow.js”, “bg”: true}}

Slide

Slide in from the edge of the screen. The `direction` prop controls which edge of the screen the transition starts from.

The Transition component's `mountOnEnter` prop prevents the child component from being mounted until `in` is `true`. This prevents the relatively positioned component from scrolling into view from its off-screen position. Similarly, the `unmountOnExit` prop removes the component from the DOM after it has been transition off-screen.

{{"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.

{{"demo": "SlideFromContainer.js"}}

Zoom

Expand outwards from the center of the child element.

This example also demonstrates how to delay the enter transition.

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

Child requirement

- **Forward the style**: To better support server rendering, MUI provides a `style` prop to the children of some transition components (Fade, Grow, Zoom, Slide). The `style` prop must be applied to the DOM for the animation to work as expected.
- **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).

```
// The `props` object contains a `style` prop.
// You need to provide it to the `div` element as shown here.
const MyComponent = React.forwardRef((props, ref) {
  return (
    <div ref={ref} {...props}>
      Fade
    </div>
  );
})

export default Main() {
  return (
    <Fade>
```

```
      {/* MyComponent must the only child */}
      <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”}}

TransitionComponent prop

Some MUI components use these transitions internally. These accept a `TransitionComponent` prop to customize the default transition. You can use any of the above components or your own. It should respect the following conditions:

- Accepts an `in` prop. This corresponds to the open/close state.
- Call the `onEnter` callback prop when the enter transition starts.
- Call the `onExited` callback prop when the exit transition is completed. These two callbacks allow to unmount the children when in a closed state and fully transitioned.

For more information on creating a custom transition, visit the *react-transition-group* `Transition` documentation. You can also visit the dedicated sections of some of the components:

- 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.