Tooltip

Tooltips display informative text when users hover over, focus on, or tap an element.

When activated, Tooltips display a text label identifying an element, such as a description of its function.

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

Basic tooltip

{{"demo": "BasicTooltip.js"}}

Positioned tooltips

The `Tooltip` has 12 **placements** choice. They don't have directional arrows; instead, they rely on motion emanating from the source to convey direction.

{{"demo": "PositionedTooltips.js"}}

Customization

Here are some examples of customizing the component. You can learn more about this in the overrides documentation page.

{{"demo": "CustomizedTooltips.js"}}

Arrow tooltips

You can use the `arrow` prop to give your tooltip an arrow indicating which element it refers to.

{{"demo": "ArrowTooltips.js"}}

Custom child element

The tooltip needs to apply DOM event listeners to its child element. If the child is a custom React element, you need to make sure that it spreads its props to the underlying DOM element.

```
const MyComponent = React.forwardRef(function MyComponent(props, ref) {
  //  Spread the props to the underlying DOM element.
  return <div {...props} ref={ref}>Bin</div>
});

// ...

<Tooltip title="Delete">
  <MyComponent>
</Tooltip>
```

You can find a similar concept in the wrapping components guide.

Triggers

You can define the types of events that cause a tooltip to show.

The touch action requires a long press due to the `enterTouchDelay` prop being set to `700` ms by default.

{{"demo": "TriggersTooltips.js"}}

Controlled tooltips

You can use the `open`, `onOpen` and `onClose` props to control the behavior of the tooltip.

{{"demo": "ControlledTooltips.js"}}

Variable width

The `Tooltip` wraps long text by default to make it readable.

{{"demo": "VariableWidth.js"}}

Interactive

Tooltips are interactive by default (to pass WCAG 2.1 success criterion 1.4.13). It won't close when the user hovers over the tooltip before the `leaveDelay` is expired. You can disable this behavior (thus failing the success criterion which is required to reach level AA) by passing `disableInteractive`.

{{"demo": "NonInteractiveTooltips.js"}}

Disabled elements

By default disabled elements like `<button>` do not trigger user interactions so a `Tooltip` will not activate on normal events like hover. To accommodate disabled elements, add a simple wrapper element, such as a `span`.

> ⚠️ *In order to work with Safari, you need at least one display block or flex item below the tooltip wrapper.*

{{"demo": "DisabledTooltips.js"}}

> *If you're not wrapping a MUI component that inherits from `ButtonBase`, for instance, a native `<button>` element, you should also add the CSS property pointer-events: none; to your element when disabled:*

```
<Tooltip title="You don't have permission to do this">
  <span>
    <button disabled={disabled} style={disabled ? { pointerEvents: 'none' } : {}}>
      A disabled button
    </button>
  </span>
</Tooltip>
```

Transitions

Use a different transition.

{{"demo": "TransitionsTooltips.js"}}

Follow cursor

You can enable the tooltip to follow the cursor by setting `followCursor={true}`.

{{"demo": "FollowCursorTooltips.js"}}

Virtual element

In the event you need to implement a custom placement, you can use the `anchorEl` prop: The value of the `anchorEl` prop can be a reference to a fake DOM element. You need to create an object shaped like the `VirtualElement`.

{{"demo": "AnchorElTooltips.js"}}

Showing and hiding

The tooltip is normally shown immediately when the user's mouse hovers over the element, and hides immediately when the user's mouse leaves. A delay in showing or hiding the tooltip can be added through the `enterDelay` and `leaveDelay` props, as shown in the Controlled Tooltips demo above.

On mobile, the tooltip is displayed when the user longpresses the element and hides after a delay of 1500ms. You can disable this feature with the `disableTouchListener` prop.

{{"demo": "DelayTooltips.js"}}

Accessibility

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

By default, the tooltip only labels its child element. This is notably different from `title` which can either label **or** describe its child depending on whether the child already has a label. For example, in:

```
<button title="some more information">A button</button>
```

the `title` acts as an accessible description. If you want the tooltip to act as an accessible description you can pass `describeChild`. Note that you shouldn't use `describeChild` if the tooltip provides the only visual label. Otherwise, the child would have no accessible name and the tooltip would violate success criterion 2.5.3 in WCAG 2.1.

{{"demo": "AccessibilityTooltips.js"}}