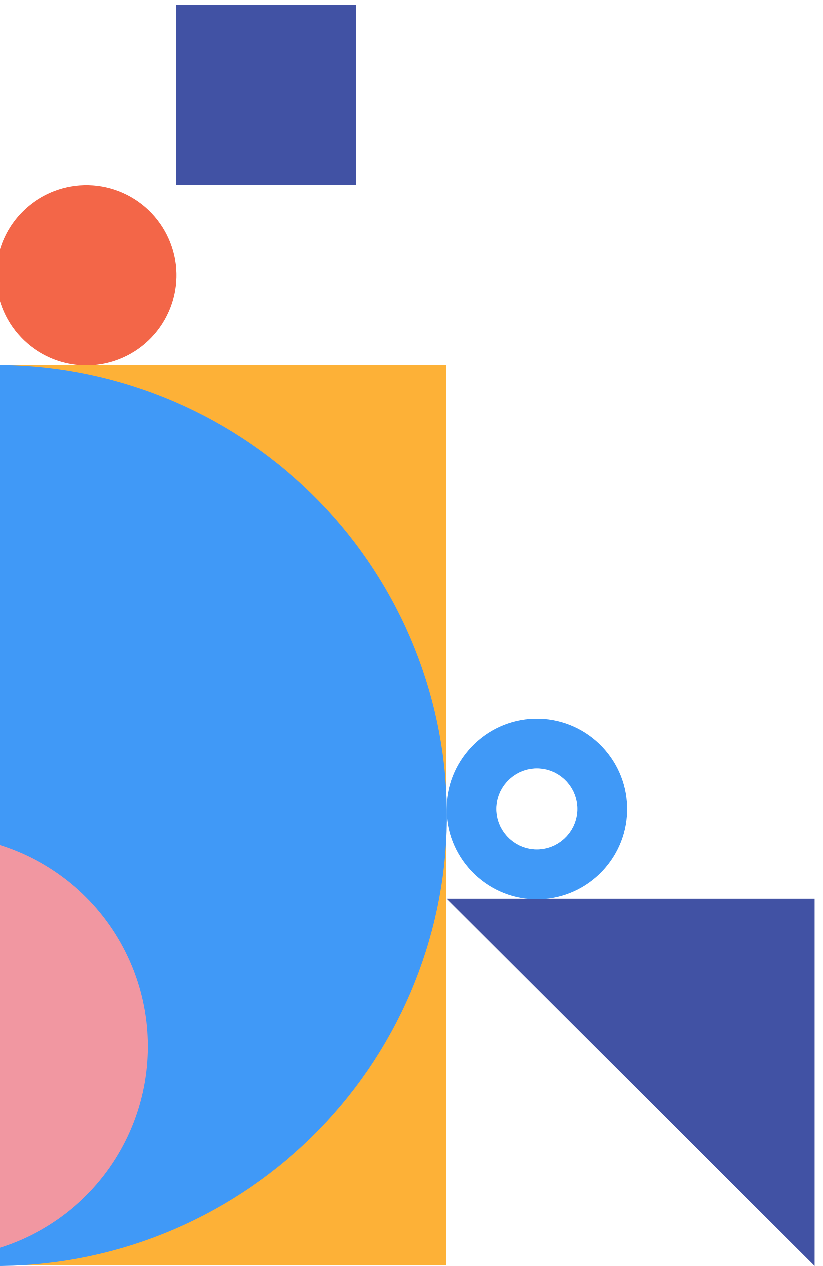


Animating Components

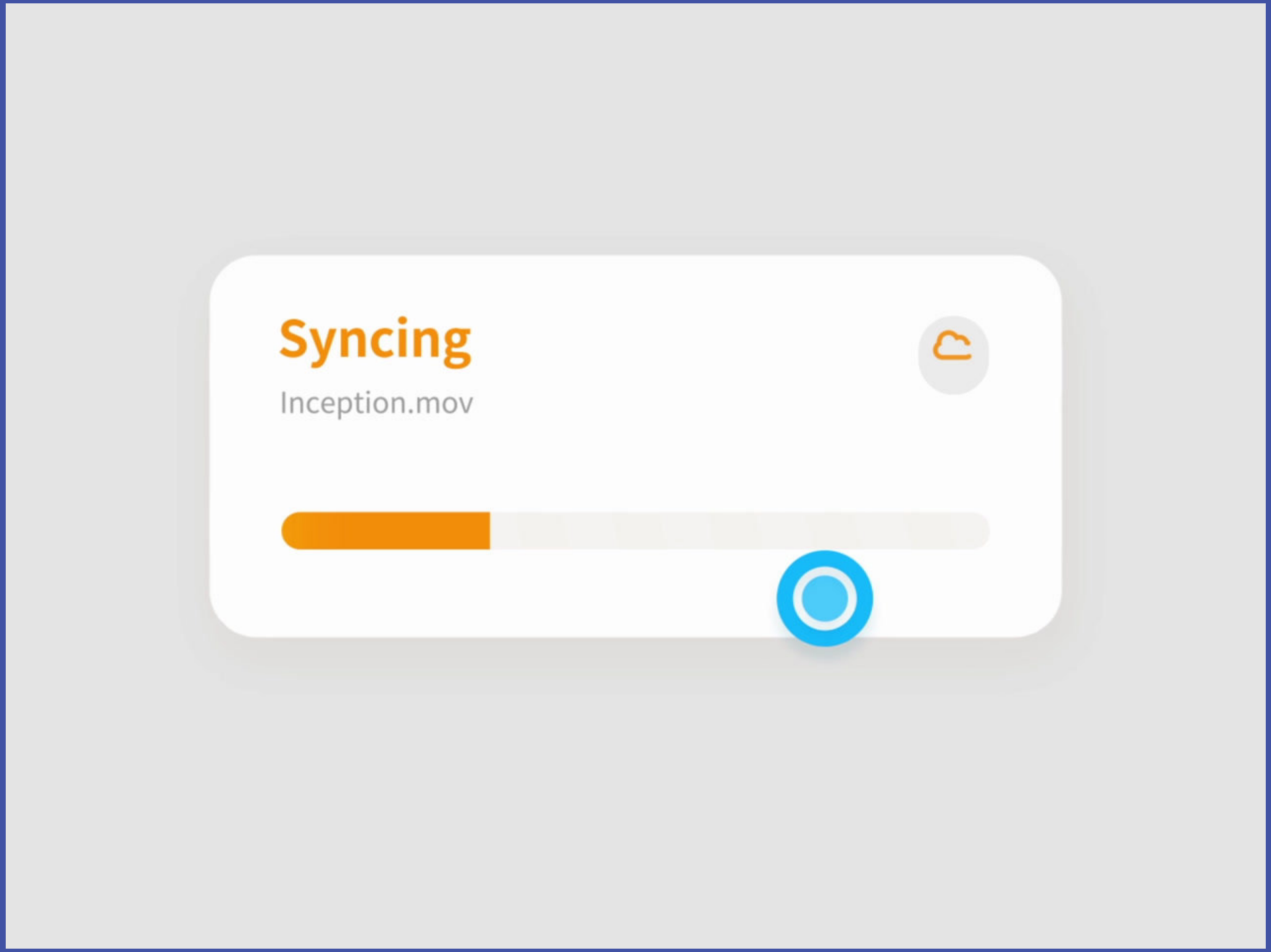
Now that we've built our components, let's animate them
with react-spring.

Microinteractions

Micro-interactions are small animations whose purpose is to delight the user by providing feedback in regards to a task or inform the user about the status of a process or task.

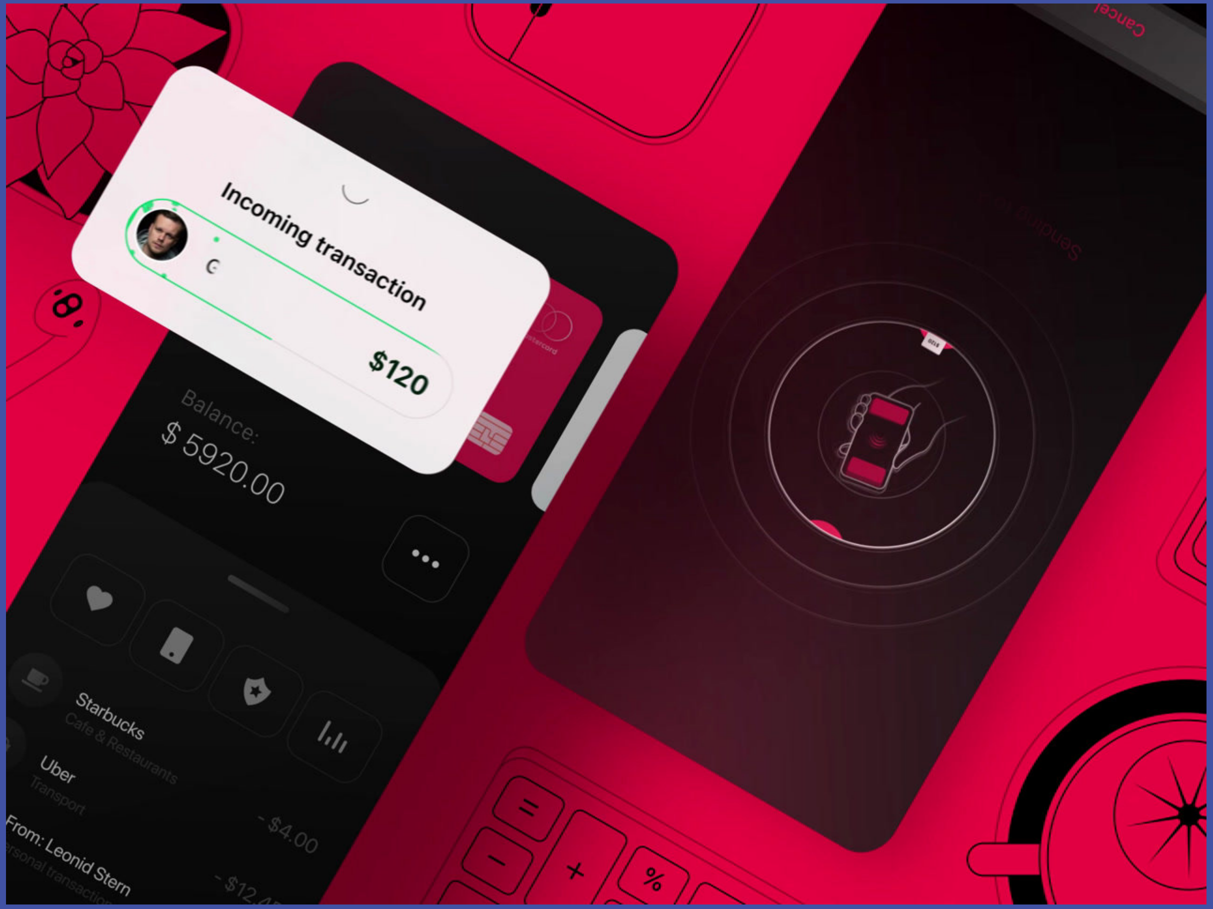
Perceived performance

We can alter our user's sense of time with animations and this can work in our favor if our performance budget needs some refactoring.



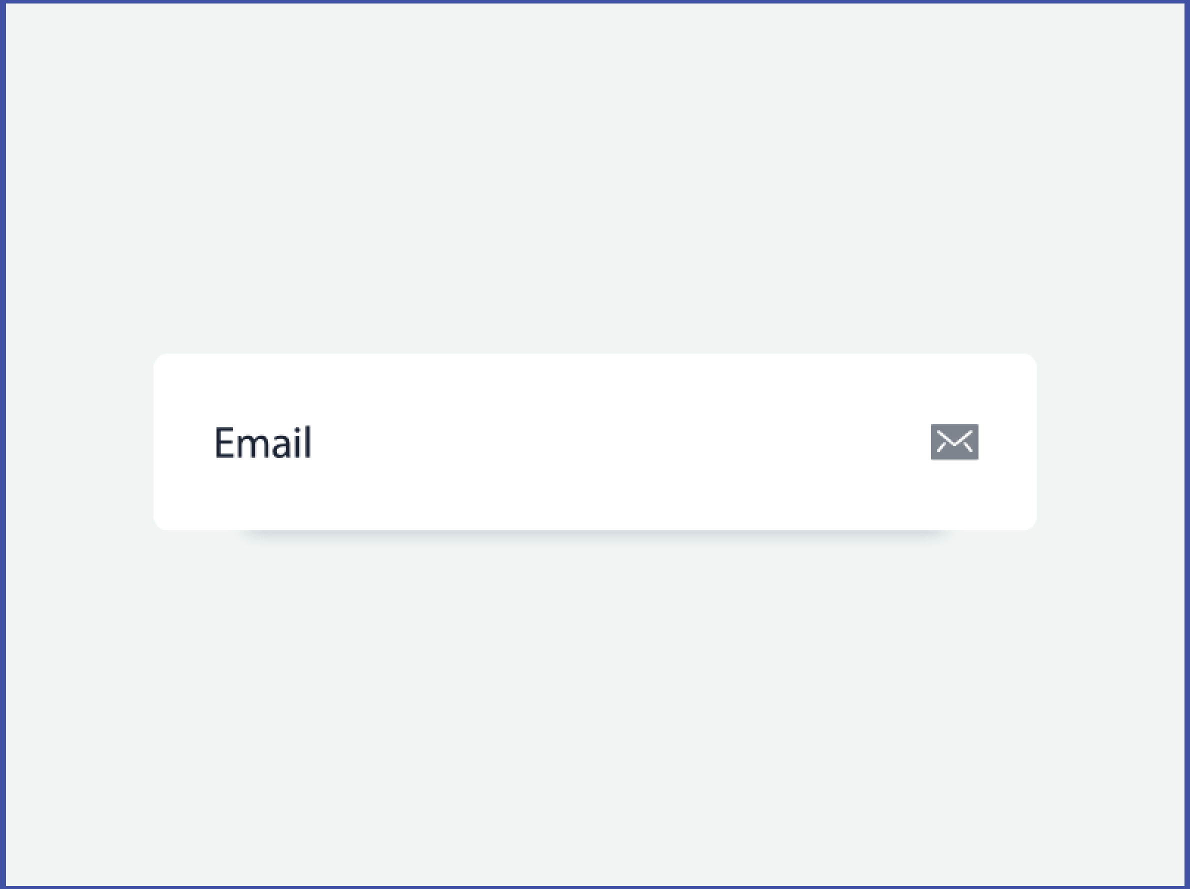
Task status

As a user's request is processing or as their data is loading, we can use a micro-interaction to inform them of its status.



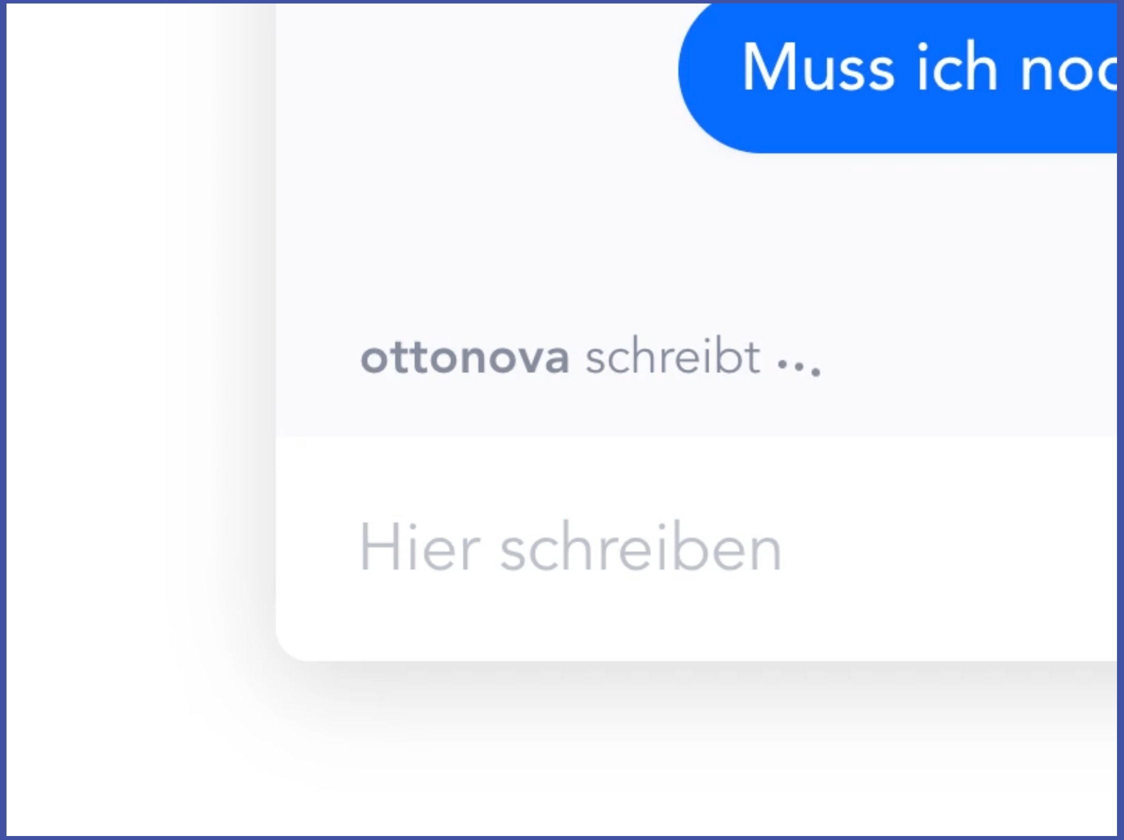
State change

If a user is filling out a form and incorrectly enters their password, we can use micro-interactions to illustrate that this form needs to be fixed prior to submission.



Draw attention

Using micro-interactions to capture a user's attention and indicate that there is something of importance is a useful tool for on-boarding or to indicate someone is typing.

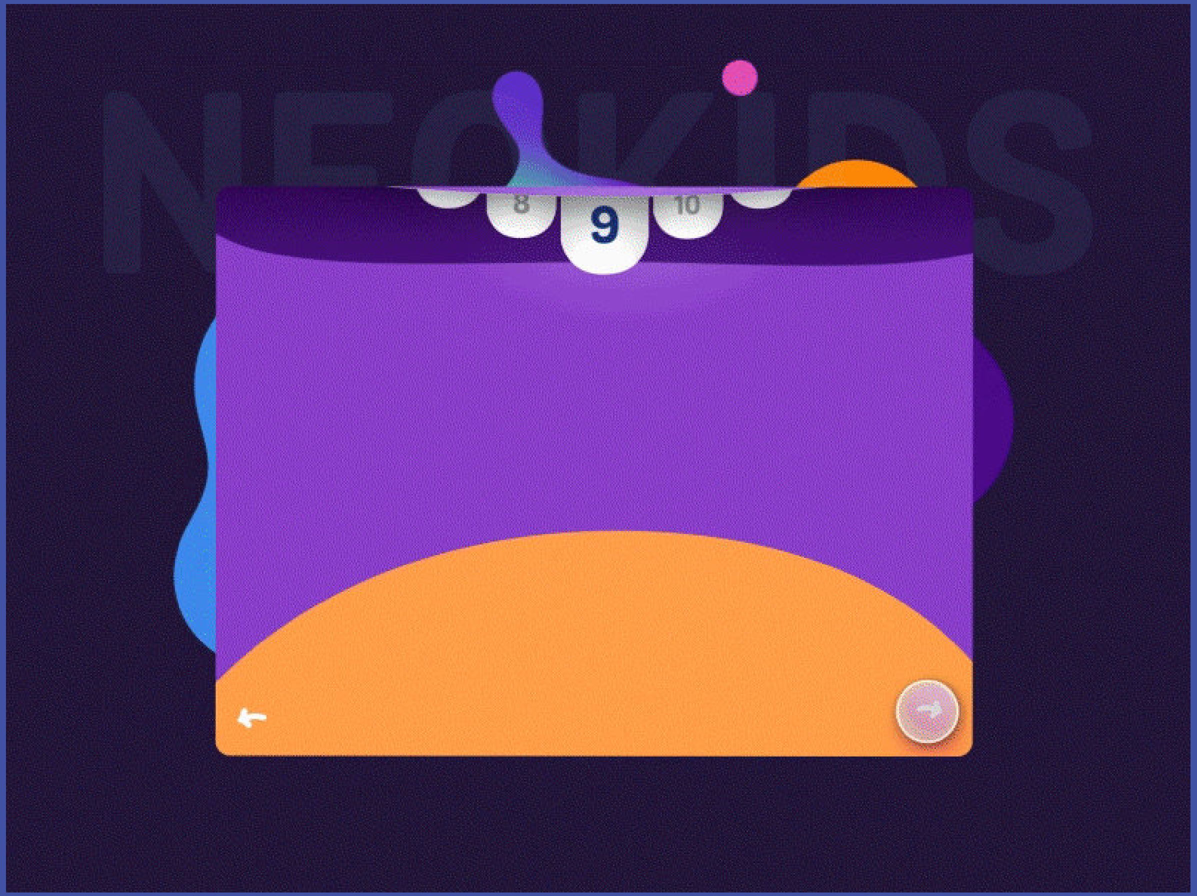


Create habits

Social media applications are really good at getting their uses to form habits, and they do so with micro-interactions.

Delight users

Micro-interactions can bring joy to our users by enhancing their experience.



Tips for building animations

Tips for building animations

ACCESSIBILITY

Your animations must be accessible.

Tips for building animations

ACCESSIBILITY

Your animations must be accessible.

INTENTIONALITY

Be intentional with the placement of your animations.

Tips for building animations

ACCESSIBILITY

Your animations must be accessible.

RELATABILITY

Make your animations feel as though they're part of the real world.

INTENTIONALITY

Be intentional with the placement of your animations.

Tips for building animations

ACCESSIBILITY

Your animations must be accessible.

RELATABILITY

Make your animations feel as though they're part of the real world.

INTENTIONALITY

Be intentional with the placement of your animations.

PERFORMANCE

Never make your users wait for an animation.

react-spring

react-spring

react-spring is a hooks-based and a physics-based animation library, and allows you to create complex animations.

useSpring

Allows you to transition an element's CSS properties on enter and exit as well as relative to your React state.

```
const animation = useSpring({
  from: { opacity: 0, transform: `translateY(-200%)` },
  to: { opacity: 1, transform: `translateY(0)` }
});
```

useSpring

Allows you to transition an element's CSS properties on enter and exit as well as relative to your React state.

```
const animation = useSpring({
  to: {
    opacity: 1,
    transform: `translateY(0%)`
  }
});
```

useSpring

Allows you to transition an element's CSS properties on enter and exit as well as relative to your React state.

```
const animation = useSpring({
  opacity: 1,
  transform: `translateY(0%)`
});
```

useSpring

Allows you to transition an element’s CSS properties on enter and exit as well as relative to your React state.

```
const [showModal, setShowModal] = useState(false);
const animation = useSpring({
    opacity: showModal ? 1 : 0,
    transform: showModal ? `translateY(0%)` : `translateY(-200%)
});
```

useSpring

Allows you to transition an element's CSS properties on enter and exit as well as relative to your React state.

```
<animated.div></animated.div>
<animated.h1></animated.h1>
```

useSpring

Allows you to transition an element's CSS properties on enter and exit as well as relative to your React state.

```
<animated.div style={animation}></animated.div>
```

```
import React, { useState } from 'react';
import { animated, useSpring } from 'react-spring';

const Modal = () => {
    const [showModal, setShowModal] = useState(false);
    const animation = useSpring({
        opacity: showModal ? 1 : 0,
        transform: showModal ? `translateY(0%)` : `translateY(-200%)
    });

    return <animated.div style={animation}>...</animated.div>
}

export default Modal;
```

Fade in

http://bit.ly/3cHj8gC

useTransition

Transition an array of elements.

```
const [items, set] = useState([...])
const transitions = useTransition(items, item => item.key, {
    from: { opacity: 0 },
    enter: { opacity: 1 },
    leave: { opacity: 0 }
})

return transitions.map(({ item, props, key }) =>
    <animated.div key={key} style={props}>{item.text}</animated.div>
)
```

useTransition

Toggle between two different elements.

```
const [toggle, set] = useState(false);
const transitions = useTransition(toggle, null, {
  from: { position: "absolute", opacity: 0 },
  enter: { opacity: 1 },
  leave: { opacity: 0 }
});
return transitions.map(({ item, key, props }) =>
  item ? (
    <animated.div style={props}>Hello</animated.div>
  ) : (
    <animated.div style={props}>Goodbye</animated.div>
  )
);
```

useTransition

Mount and un-mount one element from the DOM.

```
const [show, set] = useState(false);
const transitions = useTransition(show, null, {
  from: { position: "absolute", opacity: 0 },
  enter: { opacity: 1 },
  leave: { opacity: 0 }
});
return transitions.map(
  ({ item, key, props }) =>
    item && (
      <animated.div key={key} style={props}>
        I'm mounted!
      </animated.div>
    )
);
```

Emoji carousel

http://bit.ly/3cHj8gC