Emil Kowalski
Design Engineer

One mistake I made early in my career was thinking that design, and animations in particular, are an art you either “get” or you don’t.

Everything beautiful felt like magic, and that magic felt completely out of reach.

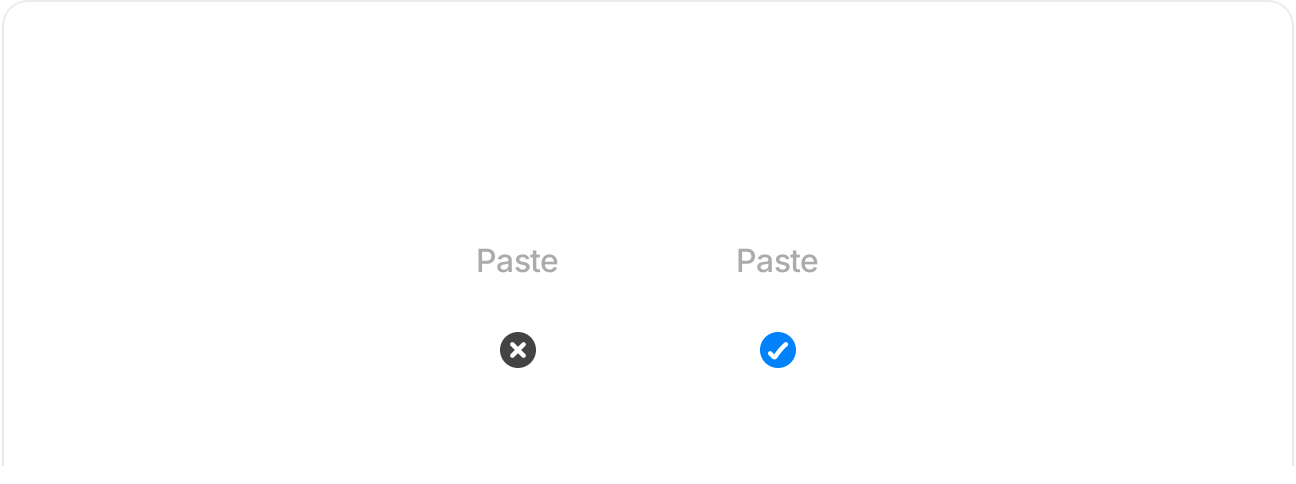
While it certainly requires developing good intuition, there are loads of tricks you can use to make your animations feel better without having to learn that “magic”.

Here are seven simple ideas you can use to improve your animations today.

1. Scale your buttons

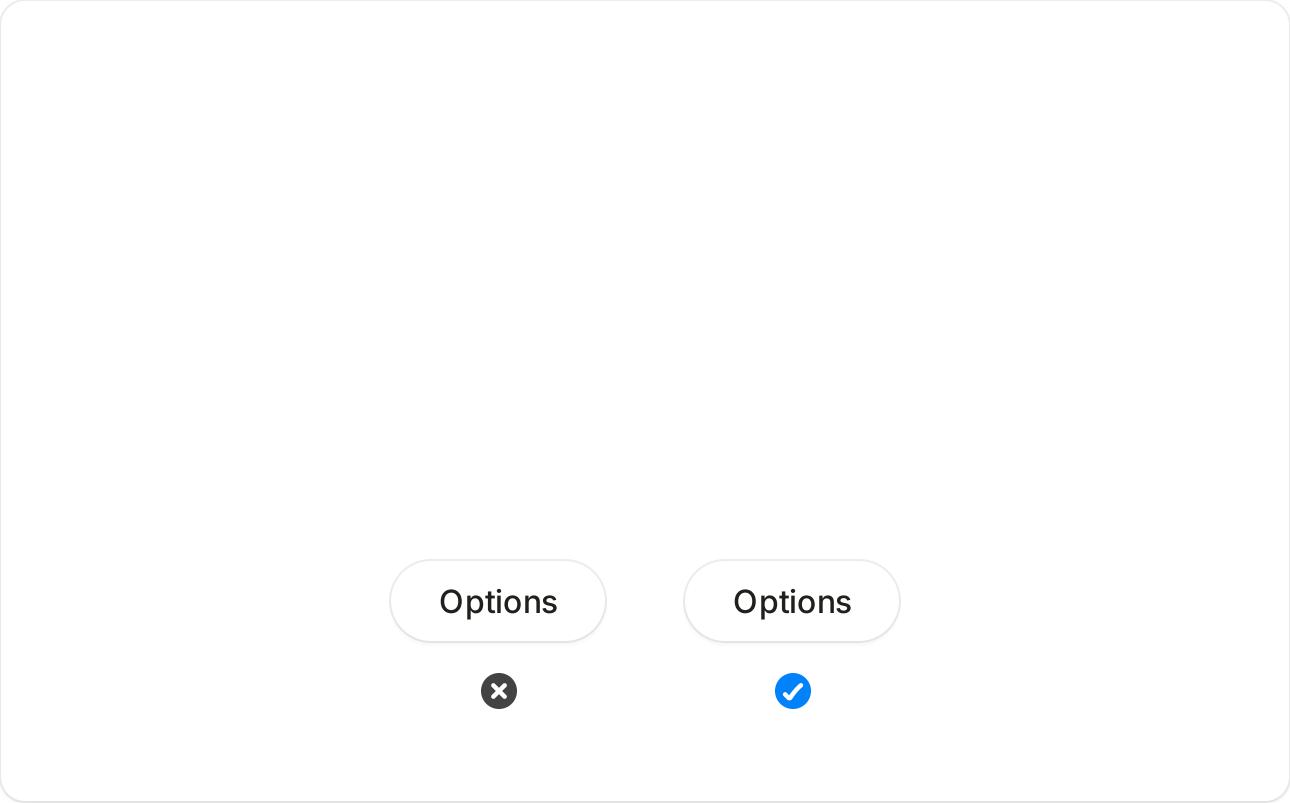
The interface should feel as if it’s listening to the user. You should aim to give the user feedback on all of their actions as soon as possible. Submitting a form should show a loading state, copy to clipboard action should show a success state.

One easy way to make your interface feel instantly more responsive is to add a subtle scale down effect when a button is pressed. A scale of `0.97` on the `:active` pseudo-class should do the job:



2. Don’t animate from scale(0)

Elements that animate from `scale(0)` can make an animation feel off. Try animating from a higher initial scale instead (0.9+). It makes the movement feel more gentle, natural, and elegant.



I’m using an initial scale of 0.93 here.

`scale(0)` feels wrong because it looks like the element comes out of nowhere. A higher initial value resembles the real world more. Just like a balloon, even when deflated it has a visible shape, it never disappears completely.

3. Don’t delay subsequent tooltips

Tooltips should have a delay before appearing to prevent accidental activation. Once a tooltip is open, hovering over other tooltips should open them with no delay and no animation. This feels faster without defeating the purpose of the initial delay.

Hover over one of the icons to see a tooltip, then quickly hover over another icon within the same group.

Radix and Base UI, two unstyled component libraries, skip the delay once a tooltip is shown. Base UI allows you to skip the animation as well. To do that you'll need to target the `data-instant` attribute and set the transition duration to `0ms`.

Here's how the styles would look like to achieve this:

```
.tooltip {
  transition:
    transform 0.125s ease-out,
    opacity 0.125s ease-out;
  transform-origin: var(--transform-origin);

  &[data-starting-style],
  &[data-ending-style] {
    opacity: 0;
    transform: scale(0.97);
  }

  /** This takes care of disabling subsequent animations */
  &[data-instant] {
    transition-duration: 0ms;
  }
}
```

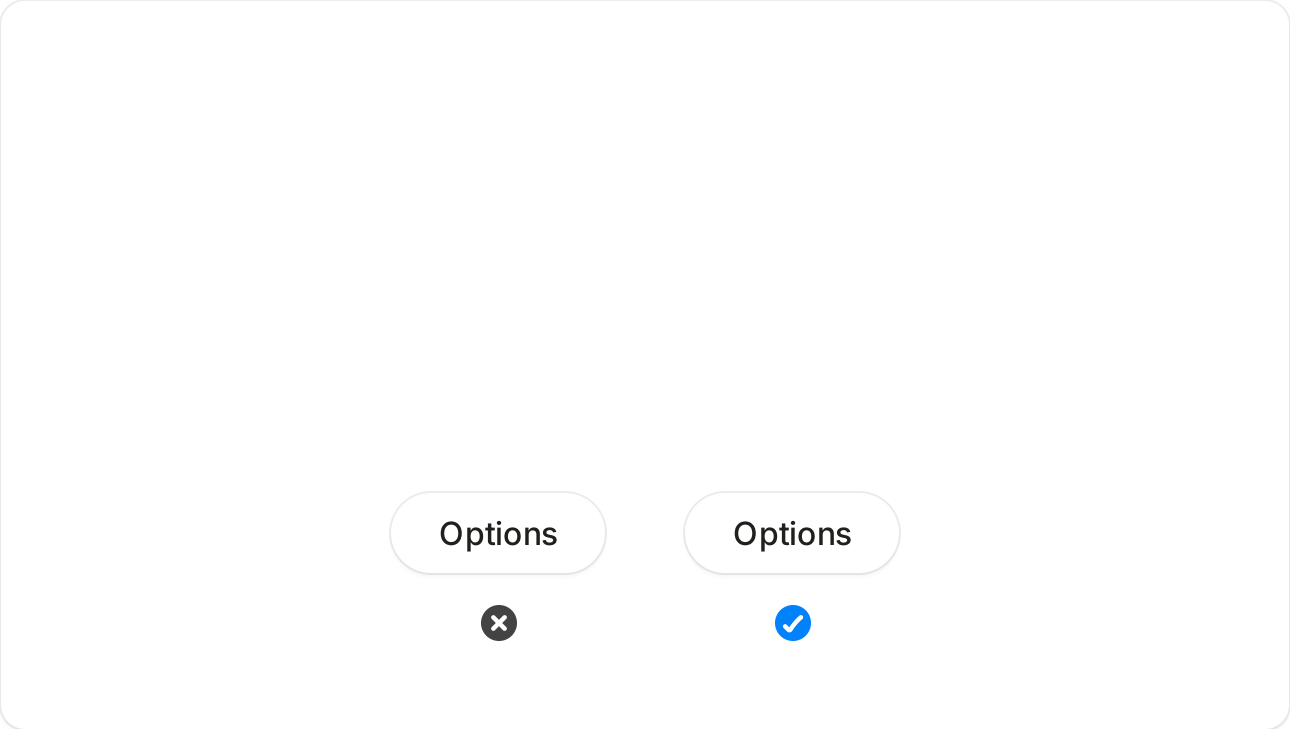
4. Choose the right easing

Easing, which describes the rate at which something changes over a period of time, is the most important part of any animation.

It can make a bad animation look great, and a great animation look bad.

If you are animating something that is entering or exiting the screen, use `ease-out`. It accelerates at the beginning which gives the user a feeling of responsiveness.

The dropdown on the left below uses `ease-in`, which starts slow. The one on the right uses `ease-out`, which starts fast, making the animation feel faster too.



The duration for these dropdowns is identical, `300ms` (you can inspect the code to verify this), but the `ease-in` one feels slower. That's the difference easing makes.

You could decrease the duration to make `ease-in` work, but this easing is just not made for UI animations. It speeds up at the *end*, which is the opposite of what we want.

Here's a visual representation of both curves, blue is `ease-out`. Notice how much faster it moves at the beginning, that's what we want for our animations.

Replay the animation by pressing on the refresh icon in the bottom right corner.

Most of the animations you've seen so far also use custom easing curves to make them feel better.

The built-in easing curves in CSS are usually not strong enough, which is why I almost never use them. Take a look at the example below where we compare two versions of the `ease-in-out` curve:

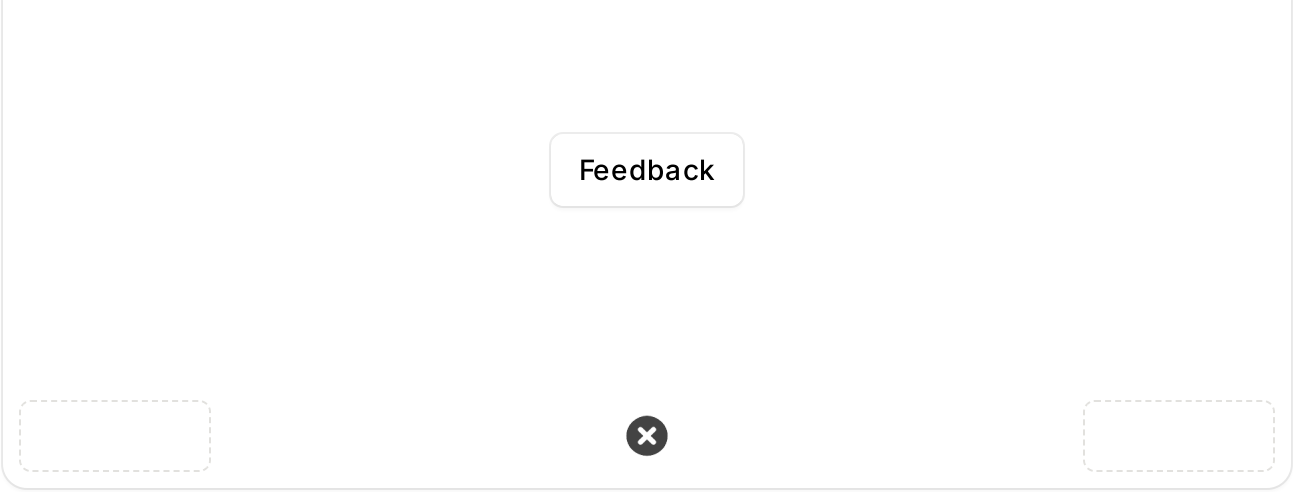
Custom easing feels more energetic.

You can still apply the easing principles, just use a custom `ease-out` for more impact. There are plenty of sites that offer custom variations of all easings, here's one I recommend: easings.co.

5. Make your animations origin-aware

A way to make your popovers feel better is to make them origin-aware. They should scale in from the trigger. You'll need CSS's `transform-origin` for this, but its default value is `center`, which is wrong in most cases.

Click on the feedback button below to open the popover and see it animate from the center, close it, and press `J` to set the correct origin and open the popover again. Pressing `S` will slow the animation down so you can see the difference better.



Press J to toggle between aware and unaware origins, then click on the button.

You can also click on the dashed buttons to move the component to a different position. The difference is most noticeable when both the horizontal and vertical origins don't match, like top right.

Base UI and Radix UI support origin-aware animations through CSS variables. This means that applying these variables will set the correct origin automatically.

```
.radix {
  transform-origin: var(--radix-dropdown-menu-content-transform-origin);
}

.baseui {
  transform-origin: var(--transform-origin);
}
```

You might think that the difference is subtle, maybe you don't even notice it. To be honest, whether you or your users notice it doesn't matter that much. In the aggregate, unseen details become visible, they compound.

> "All those unseen details combine to produce something that's just stunning, like a thousand barely audible voices all singing in tune."

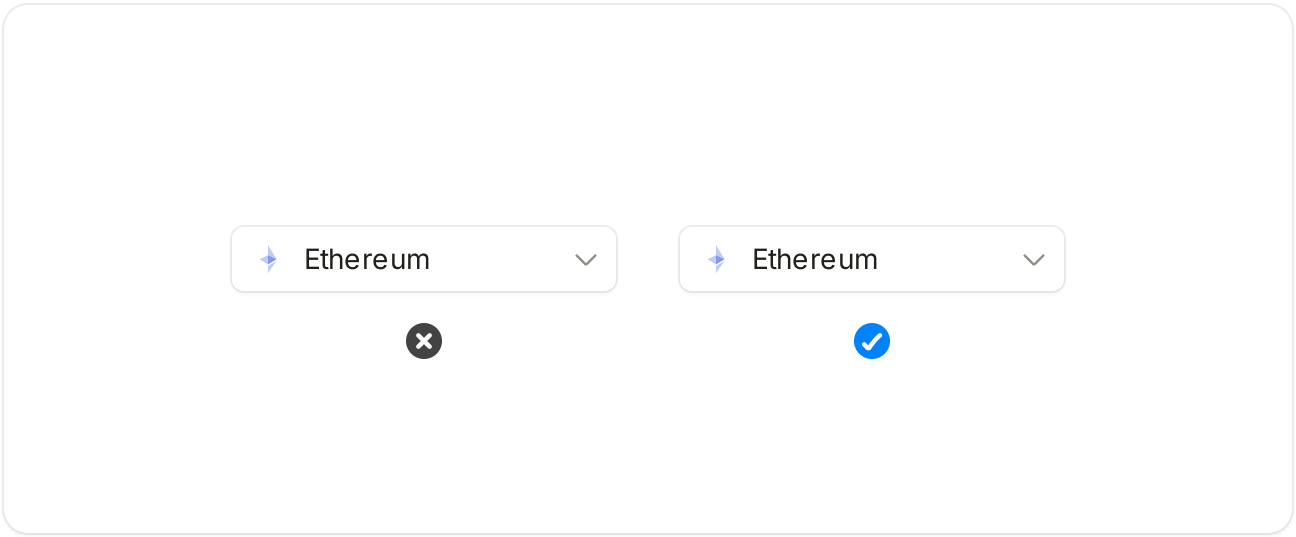
Paul Graham, Hackers and Painters

6. Keep your animations fast

A faster-spinning spinner makes the app seem to load faster, even though the load time is the same. This improves the perceived performance.

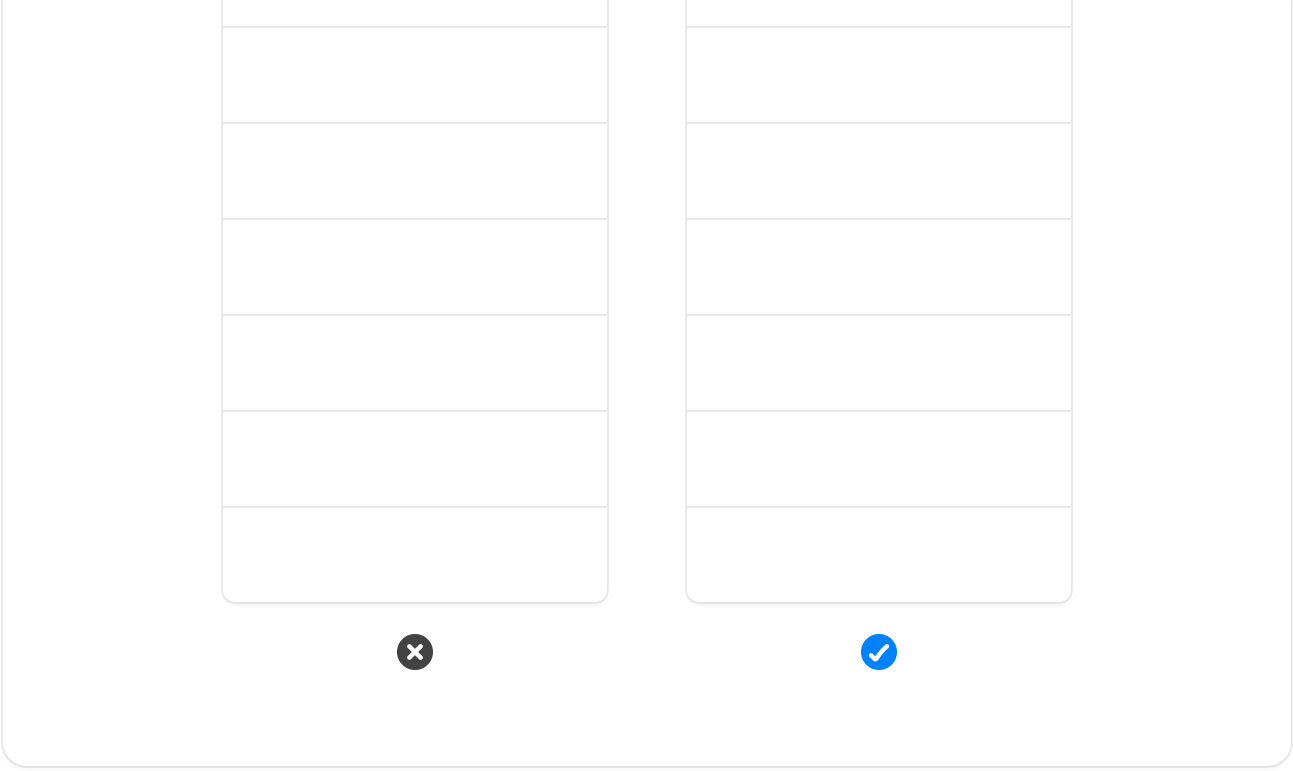
Which one works harder to load the data?

A `180ms` select animation feels more responsive than a `400ms` one:



As a rule of thumb, UI animations should generally stay under `300ms`.

Remove animations or hover interactions altogether if they are seen tens, maybe even hundreds of times a day. Instead of delighting your users, they'll quickly become annoying and make your interface feel slower.



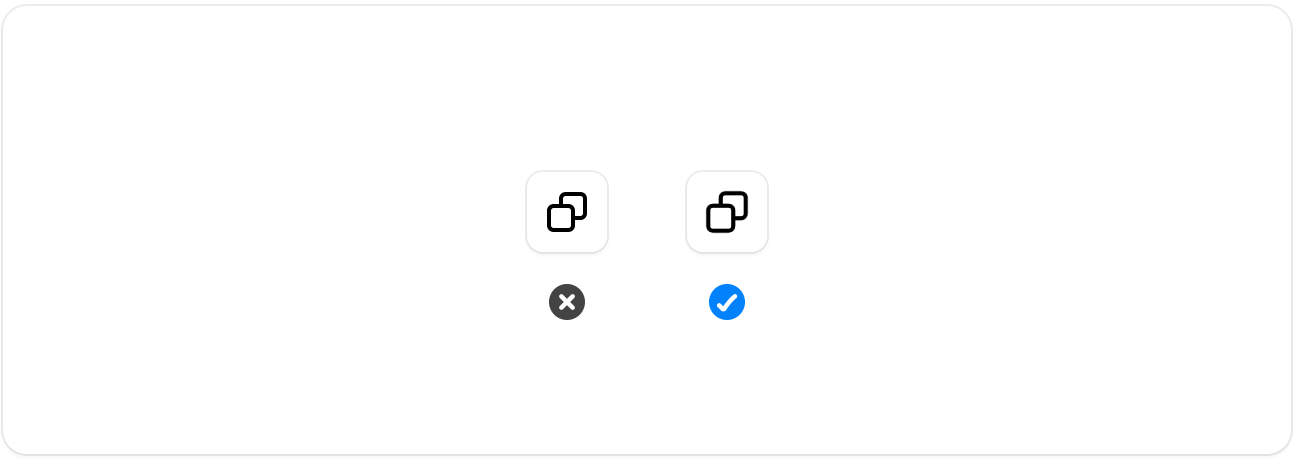
Imagine you interact with this list often during the day.

I go into this topic in more detail in the "You Don't Need Animations" post.

7. Use blur when nothing else works

If you tried a few different easings and durations for your animation and something about it still feels off, try adding a bit of `filter: blur()` to mask those imperfections.

Below we have a button that simply crossfades between two states, and another one that adds `2px` of blur to that animation. It also uses tip #1 to scale the button down to `0.97` when pressed. Notice how much more pleasing the second button feels.

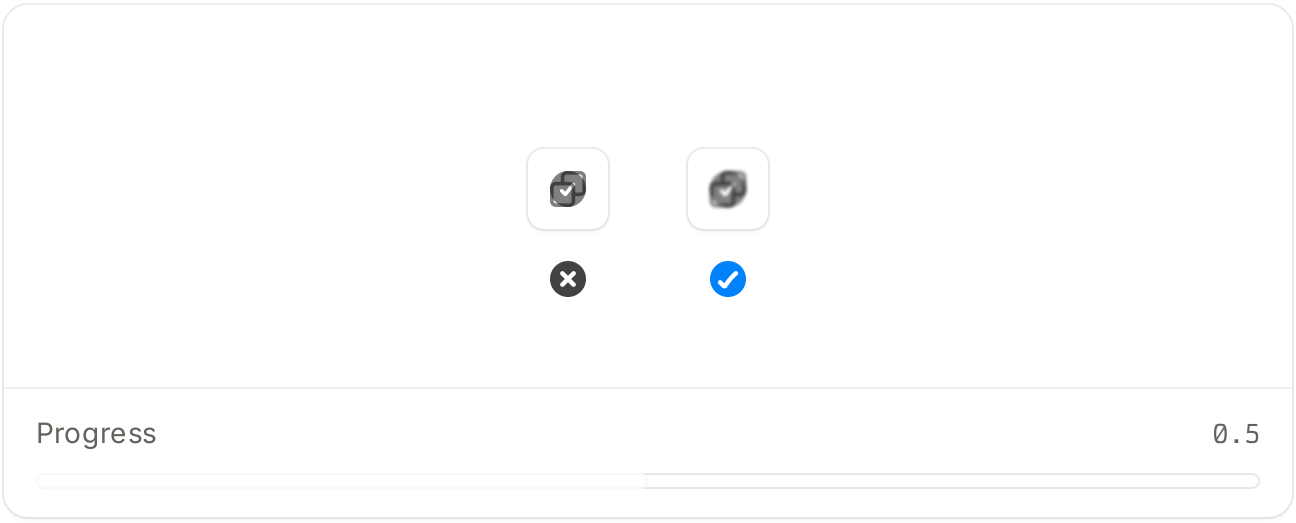


Click the button to toggle between two states.

Blur works here because it bridges the visual gap between the old and new states. Without it, you see two distinct objects, which feels less natural.

It tricks the eye into seeing a smooth transition by blending the two states together.

Notice how much more distinct the two states are without blur:



Scrub through the timeline to see the animation at different stages.

Continue learning

These tips will level up your animations, but this article only scratches the surface of how an interface can be improved through motion.

In animations.dev we go way deeper into the theory and practice of great animations. There are videos and interactive components that help you understand the theory, and exercises for you to practice building animations that feel right.

The last enrollment of this year opens in less than a week for **10 days**. You can sign up for the waitlist using the link below to get a **free course preview** to see if it's a good fit for you:

→ Check out animations.dev