Setting values and watching the DOM update automatically is cool. Know what's even cooler? *Tweening* those values. Svelte includes tools to help you build slick user interfaces that use animation to communicate changes.

Let's start by changing the `progress` store to a `tweened` value:

```
<script>
    import { tweened } from 'svelte/motion';

    const progress = tweened(0);
</script>
```

Clicking the buttons causes the progress bar to animate to its new value. It's a bit robotic and unsatisfying though. We need to add an easing function:

```
<script>
    import { tweened } from 'svelte/motion';
    import { cubicOut } from 'svelte/easing';

    const progress = tweened(0, {
        duration: 400,
        easing: cubicOut
    });
</script>
```

> *The `svelte/easing` module contains the Penner easing equations, or you can supply your own `p => t` function where `p` and `t` are both values between 0 and 1.*

The full set of options available to `tweened`:

- `delay` — milliseconds before the tween starts
- `duration` — either the duration of the tween in milliseconds, or a `(from, to) => milliseconds` function allowing you to (e.g.) specify longer tweens for larger changes in value
- `easing` — a `p => t` function
- `interpolate` — a custom `(from, to) => t => value` function for interpolating between arbitrary values. By default, Svelte will interpolate between numbers, dates, and identically-shaped arrays and objects (as long as they only contain numbers and dates or other valid arrays and objects). If you want to interpolate (for example) colour strings or transformation matrices, supply a custom interpolator

You can also pass these options to `progress.set` and `progress.update` as a second argument, in which case they will override the defaults. The `set` and `update` methods both return a promise that resolves when the tween completes.