The `tick` function is unlike other lifecycle functions in that you can call it any time, not just when the component first initialises. It returns a promise that resolves as soon as any pending state changes have been applied to the DOM (or immediately, if there are no pending state changes).

When you update component state in Svelte, it doesn't update the DOM immediately. Instead, it waits until the next *microtask* to see if there are any other changes that need to be applied, including in other components. Doing so avoids unnecessary work and allows the browser to batch things more effectively.

You can see that behaviour in this example. Select a range of text and hit the tab key. Because the `<textarea>` value changes, the current selection is cleared and the cursor jumps, annoyingly, to the end. We can fix this by importing `tick` ...

```
import { tick } from 'svelte';
```

...and running it immediately before we set `this.selectionStart` and `this.selectionEnd` at the end of `handleKeydown` :

```
await tick();
this.selectionStart = selectionStart;
this.selectionEnd = selectionEnd;
```