Media elements

The `<audio>` and `<video>` elements have several properties that you can bind to. This example demonstrates a few of them.

On line 62, add `currentTime={time}`, `duration` and `paused` bindings:

```
<video
	poster="https://sveltejs.github.io/assets/caminandes-llamigos.jpg"
	src="https://sveltejs.github.io/assets/caminandes-llamigos.mp4"
	on:mousemove={handleMove}
	on:touchmove|preventDefault={handleMove}
	on:mousedown={handleMousedown}
	on:mouseup={handleMouseup}
	bind:currentTime={time}
	bind:duration
	bind:paused>
	<track kind="captions">
</video>
```

> `bind:duration` is equivalent to `bind:duration={duration}`

Now, when you click on the video, it will update `time`, `duration` and `paused` as appropriate. This means we can use them to build custom controls.

> Ordinarily on the web, you would track `currentTime` by listening for `timeupdate` events. But these events fire too infrequently, resulting in choppy UI. Svelte does better — it checks `currentTime` using `requestAnimationFrame`.

The complete set of bindings for `<audio>` and `<video>` is as follows — six *readonly* bindings...

- `duration` (readonly) — the total duration of the video, in seconds
- `buffered` (readonly) — an array of `{start, end}` objects
- `seekable` (readonly) — ditto
- `played` (readonly) — ditto
- `seeking` (readonly) — boolean
- `ended` (readonly) — boolean

...and five *two-way* bindings:

- `currentTime` — the current point in the video, in seconds
- `playbackRate` — how fast to play the video, where `1` is 'normal'
- `paused` — this one should be self-explanatory
- `volume` — a value between 0 and 1
- `muted` — a boolean value where true is muted

Videos additionally have readonly `videoWidth` and `videoHeight` bindings.