In this app, we have a `<Hoverable>` component that tracks whether the mouse is currently over it. It needs to pass that data *back* to the parent component, so that we can update the slotted contents.

For this, we use *slot props*. In `Hoverable.svelte`, pass the `hovering` value into the slot:

```
<div on:mouseenter={enter} on:mouseleave={leave}>
    <slot hovering={hovering}></slot>
</div>
```

> *Remember you can also use the `{hovering}` shorthand, if you prefer.*

Then, to expose `hovering` to the contents of the `<Hoverable>` component, we use the `let` directive:

```
<Hoverable let:hovering={hovering}>
    <div class:active={hovering}>
        {#if hovering}
            <p>I am being hovered upon.</p>
        {:else}
            <p>Hover over me!</p>
        {/if}
    </div>
</Hoverable>
```

You can rename the variable, if you want — let's call it `active` in the parent component:

```
<Hoverable let:hovering={active}>
    <div class:active>
        {#if active}
            <p>I am being hovered upon.</p>
        {:else}
            <p>Hover over me!</p>
        {/if}
    </div>
</Hoverable>
```

You can have as many of these components as you like, and the slotted props will remain local to the component where they're declared.

> *Named slots can also have props; use the `let` directive on an element with a `slot="..."` attribute, instead of on the component itself.*