Testing Named Slots

In this lesson, we'll be testing Named Slots.

WE'LL COVER THE FOLLOWING

Default Slots

The unnamed slot we used in the previous lesson is called the *default slot*, but we can have multiple slots by using named slots. Let's add a header to the `MessageList.vue` component:

MessageList.vue

```
<template>
  <div>
    <header class="list-header">
      <slot name="header">
        This is a default header
      </slot>
    </header>
    <ul class="list-messages">
      <slot></slot>
    </ul>
  </div>
</template>
```

By using `<slot name="header">` we're defining another slot for the header. You can see a `This is a default header` text inside the slot which is displayed as the default content.

Then, from `App.vue` we can add a header to the `MessageList` component by using the `slot="header"` attribute:

```
<template>
```

```
  <div id="app">
    <MessageList>
      <header slot="header">
        Awesome header
      </header>
      <Message
        @message-clicked="handleMessageClick"
        :message="message"
        v-for="message in messages"
        :key="message"/>
    </MessageList>
  </div>
</template>
```

Writing a Unit Test for Named Slots

It's time to write a unit test for named slots. Testing named slots are just like testing a default slot; the same dynamics apply. So, we can start by testing that the header slot is rendered within the `<header class="list-header">` element, and that it renders a default text when no header slot is passed by. In `MessageList.test.js`:

MessageList.test.js

```
it("Header slot renders a default header text", () => {
  const header = cmp.find(".list-header");
  expect(header.text().trim()).toBe("This is a default header");
});
```

Things remain the same but now, checking the default content gets replaced when we mock the header slot:

```
it("Header slot is rendered withing .list-header", () => {
  const component = shallowMount(MessageList, {
    slots: {
      header: "<div>What an awesome header</div>"
    }
  });

  const header = component.find(".list-header");
  expect(header.text().trim()).toBe("What an awesome header");
});
```

Notice that the header slot used in this last test is wrapped in a `<div>`. It's important the slots are wrapped in an html tag, otherwise vue-test-utils will complain.

In the next lesson, we'll learn how a slot must behave in the context of the component.