Testing Slots

In this lesson, we will be Slot Testing.

WE'LL COVER THE FOLLOWING

- Slot Testing in Action

What we want to test the most out of slots is where they end up in the component. For that, we can reuse the skills we learned in the chapter *Test Styles and Structure*.

Right now, most of the tests in `MessageList.test.js` will fail, so let's remove them all (or comment them out), and focus on slot testing.

Slot Testing in Action

One thing we can do is to make sure that the Message components end up within a `ul` element with the class `list-messages`. In order to pass slots to the `MessageList` component, we can use the `slots` property of the options object of `mount` or `shallowMount` methods. So let's create a `beforeEach` method with the following code:

```
beforeEach(() => {
  cmp = shallowMount(MessageList, {
    slots: {
      default: '<div class="fake-msg"></div>'
    }
  });
});
```

Since we just want to test if the messages are rendered, we can search for `<div class="fake-msg"></div>` as follows:

```
it("Messages are inserted in a ul.list-messages element", () => {
  const list = cmp.find("ul.list-messages");
  expect(list.findAll(".fake-msg").length).toBe(1);
  expect(list.findAll(".fake-msg").length).toBe(1);
});
```

That should be ok to go. The slots option also accepts a component declaration and even an array, so we could write:

```
import AnyComponent from "anycomponent";

shallowMount(MessageList, {
  slots: {
    default: AnyComponent // or [AnyComponent, AnyComponent]
  }
});
```

The problem with it is that it is very limited; for example, you cannot override props and we need that for the `Message` component since it has a required property. This will affect the cases where you really need to test slots with the expected components. For example, if you want to make sure that `MessageList` expects only `Message` components as slots, that's on track and at some point, it will land in `vue-test-utils`.

As a workaround, we can accomplish that by using a render function. So we can rewrite the test to be more specific:

```
beforeEach(() => {
  const messageWrapper = {
    render(h) {
      return h(Message, { props: { message: "hey" } });
    }
  };

  cmp = shallowMount(MessageList, {
    slots: {
      default: messageWrapper
    }
  });
});

it("Messages are inserted in a MessageList component", () => {
  const list = cmp.find(MessageList);
  expect(list.find(Message).isVueInstance()).toBe(true);
});
```

Now that we are familiar with slot testing, let us explore Named Slots in the next lesson.