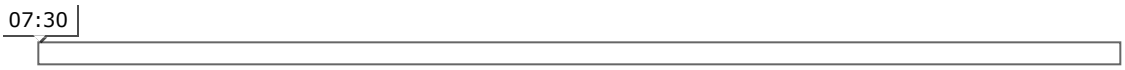
Watch: ✨ **Testeando lo que de verdad importa**



Puedes encontrar el código de este vídeo en GitHub.

En este vídeo vamos a ver un ejemplo de test que nos mostrará que podemos mejorar nuestros tests progresivamente, sin necesidad de hacer el salto a Testing Library.

Tenemos un componente que nos comunica que ha habido un error, con un botón que nos hará *toggle* para mostrar más detalles*.* Los detalles los renderizará otro componente.

Así vemos que nuestro caso de uso empieza en el componente `ErrorToggle`. En el primer test que nos encontramos, estamos aplicando una serie de malas prácticas:

```
describe("Error Toggle", () => {
  it("should toggle open property when button is clicked", async
() => {
    const wrapper = shallowMount(ErrorToggle);
    const button = wrapper.find("#toggle-message");

    button.trigger("click");
    expect(wrapper.vm.open).toBe(true);

    button.trigger("click");
    expect(wrapper.vm.open).toBe(false);
  });
});
```

Vemos el uso de `shallowMount`, que como recordamos lo que hará es doblar los componentes hijos, en este caso, `ErrorMessage`. También vemos que en lugar de testear si se ve el mensaje o no (cosa que no podemos hacer ya que el componente hijo será un *stub*), accedemos a la propiedad `open` dentro de la instancia del componente Vue. Este test no nos da confianza, ya que podemos romper la funcionalidad fácilmente y que el test siga pasando.

Podemos mejorar este test con algunos cambios:

```
describe("Error Toggle", () => {
  it("should toggle contact message when button is clicked",
async () => {
    const wrapper = mount(ErrorToggle);
    const button = wrapper.find('[data-test-id="toggle-
message"]');
    button.trigger("click");

    await nextTick();

    let message = wrapper.find('[data-test-id="message"]');
    expect(message.exists()).toBe(true);
    expect(message.text()).toMatch(/contact us/i);

    button.trigger("click");

    await nextTick();

    message = wrapper.find('[data-test-id="message"]');
    expect(message.exists()).toBe(false);
```

```
  });
});
```

Al usar `mount`, podemos acceder al texto del mensaje para comprobar si el toggle realmente está funcionando. También hemos cambiado la manera de encontrar los elementos para usar `data-test-id` en lugar de `id`. Este test nos da mucha más confianza que el anterior, aunque vemos cosas como `await nextTick()`, que tenemos que usar para esperar a que el DOM haya cambiado antes de hacer la aserción.

Podemos ir un paso más allá y utilizar un *page object* para encapsular determinadas interacciones en métodos más semánticos, que lo relacionen con nuestro lenguaje de negocio. Aquí vemos como quedaría el test anterior utilizando un page object:

```
describe("Error Toggle", () => {
  it("should toggle contact message when button is clicked",
async () => {
    const wrapper = mount(ErrorToggle);
    const page = new ErrorPageObject(wrapper);
    page.clickToggleButton();

    await page.wait();

    expect(page.hasErrorMessage()).toBe(true);
    expect(page.text()).toMatch(/contact us/i);

    page.clickToggleButton();

    await page.wait();

    expect(page.hasErrorMessage()).toBe(false);
  });
});
```