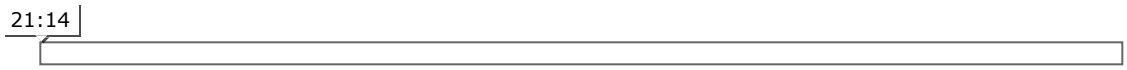
Watch: 🖌️ **Mejorar la mantenibilidad de nuestros tests con custom renderers**



Puedes encontrar el código de este vídeo en GitHub.

Siguiendo la filosofía de Testing Library hemos aprendido que es mejor no doblar nada que no suponga incrementar el tiempo o fragilidad de nuestros tests. Cuando trabajamos con librerías de gestión de estado, routing, o i18n seguiremos esta filosofía, pero son librerías que necesitamos en la mayoría de componentes, y puede ser poco práctico tener que configurar lo mismo cada vez que escribimos un test: es tedioso repetir el setup, si hay algún cambio en la configuración tendremos que cambiarlo en todos nuestros tests...

Podemos solucionar esto con custom renderers. Nos crearemos una función `render` propia en un archivo, que reciba los parámetros necesarios para configurar el componente (por ejemplo, el estado

inicial) que también reexportará toda testing library. En una aplicación Vue que use i18n i Vuex, nos quedaría algo así:

```
import { render as vtlRender } from "@testing-library/vue";
import { createI18n } from "vue-i18n";
import { store } from "../store";
import en from "../i18n/en";

export * from "@testing-library/vue"; // re-exportamos Testing
Library

export function render(component, { initialState } = {}) {
  return vtlRender(component, {
    global: {
      plugins: [
        createStore({
          ...store,
          state: {
            ...store.state,
            ...initialState // permitimos sobreescribir el
estado inicial
          }
        }),
        createI18n({
          locale: "en",
          fallbackLocale: "en",
          messages: {
            en
          }
        })
      ]
    }
  });
}
```

De esta forma, podemos importar la función `render` y demás utilidades de Testing Library como lo haríamos sin custom renderer:

```
import { render, screen, waitForElementToBeRemoved } from
"./test-utils";
```

Según las peculiaridades de cada proyecto, podemos crear diferentes renders según las librerías que queramos usar en cada componente, o

bien incluirlas todas en un solo render.