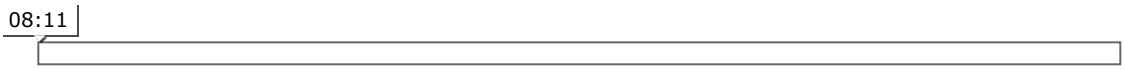
Watch: ❓ **Errores frecuentes con Testing Library**



Usar el container

Testing Library nos expone la variable `container`, que es el HTML generado por nuestro componente. Esto nos resulta útil para ciertas herramientas como Jest Axe, que realiza ciertos *checks* para comprobar la accesibilidad de ese HTML, o para realizar snapshot testing, que compara el HTML generado con la versión anterior. Pero ese HTML también nos abre la puerta a poder realizar queries directamente como `getElementById` o `querySelector`, lo que sería confiar en ciertos detalles de implementación para nuestros tests. Por esto es preferible evitar buscar elementos directamente en el `container`, y en lugar de eso usar las queries que nos proporciona Testing Library.

No usar las queries adecuadas

Testing Library nos ofrece 6 formas de realizar queries, según cómo queremos encontrar un elemento o varios:

- Para buscar de forma síncrona, usaremos `getBy...` y `getAllBy...`
- Para buscar de forma asíncrona, `findBy...` y `findAllBy...`
- Para validar que un elemento no existe, `queryBy...` y `queryAllBy...`, ya que devuelve `null` en lugar de error si no lo encuentra.

Esperar tiempo aleatorio

Antes era común forzar los tests a esperar cierto tiempo con funciones como `sleep(1000)`, para esperar a que ciertas cosas asíncronas sucedieran. Afortunadamente, el ecosistema de testing actual nos ofrece muchas formas de testear asincronía de forma más robusta, como usar `findBy` para esperar a encontrar cierto elemento, o las técnicas que veíamos en el capítulo de animaciones o el de TDD con fake timers.

Side effects en `waitFor`

La función `waitFor` de Testing Library nos permite reintentar la ejecución de cierto código hasta que este no de error. Nos puede ser útil en ciertos casos para reintentar un expect o para asegurar que una función se ha ejecutado, pero por norma general podemos conseguirlo con queries asíncronas, y nunca debemos añadir side-effects (ya que se llamarían repetidas veces).

Abusar de beforeEach

Mover ciertas partes de setup de nuestro test a `beforeEach` es tentador porque nos permite reducir duplicidad de código, pero por contra nos dificulta la lectura de los tests y hace que sean menos auto-contenidos.

Por norma general es preferible la duplicidad, y si realmente queremos subir un nivel de abstracción, mejor extraer ese código en funciones que se llamen dentro de cada test.

Enlaces relacionados:

- Common mistakes with React Testing Library
- Tipos de queries
- `waitFor`