Primeros Pasos



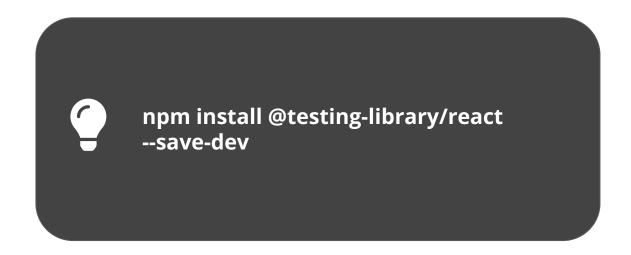
Índice

- 1. Instalación
- 2. Nuestro Primer Test

1 Instalación

Primeros Pasos

Los proyectos creados con *Create React App* tienen soporte de fábrica para React Testing Library. Pero, si ese no fuera nuestro caso, podemos añadir esta biblioteca a través de npm de esta manera:



Primeros Pasos

Una vez que hemos añadido @testing-library/react a nuestro proyecto, podemos importar render y screen, esenciales para nuestras pruebas.

```
import {render, screen} from "@testing-library/react";
```

El método render() es una función que podemos utilizar para renderizar virtualmente componentes y hacerlos disponibles en nuestras pruebas unitarias.

El objeto screen puede considerarse como una representación de la ventana del navegador. Tiene algunos otros métodos muy útiles que cubriremos en los próximos ejercicios, pero por ahora, veamos un ejemplo.



2 Nuestro Primer Test

Ejemplo práctico

Empecemos por el ejemplo más básico y necesario para comprender el funcionamiento de React Testing Library. Vamos a escribir un test para un componente muy sencillo, sobre el cual podemos probar que se renderiza bien y que tiene alguna característica determinada.

```
const Greeting = () => {
   return (<h1>Hola</h1>)
};
{}
export default Greeting;
```

Nuestro componente a probar, tiene sólo un título con la palabra "Hola" y ninguna funcionalidad. ¡Es muy simple! De hecho, sería buena idea probar que, efectivamente, renderiza la palabra "Hola". Veamos cómo sería el archivo de test para este caso:

```
import { render, screen } from '@testing-library/react';
      import { Greet } from 'components';
      describe('Greeting', () => {
            test('renderiza correctamente', () => {
                  // ① Renderizamos el componente
                  render(<Greeting />)
{}
                  //2 Visualizamos el Virtual DOM renderizado
                  screen.debug();
                  // 3 Buscamos que exista la palabra Hola en el documento
                   expect(screen.getByText(/Hola/i)).toBeInTheDocument()
            })
      })
```

Tip

Existe una dependencia que permite implementar más cantidad de matchers para facilitar los expect. En el ejemplo anterior utilizamos el matcher toBeInTheDocument el cual viene de instalar esta dependencia. Para aprovechar los matchers extra podemos copiar este comando en la terminal:

npm install --save-dev @testing-library/jest-dom

Y luego importar la librería, así: import '@testing-library/jest-dom';

Acá les facilitamos un link para conocer todos los matchers que provee

jest-dom: https://github.com/testing-library/jest-dom

Es importante notar que la salida muestra el contenido renderizado de <Greeting>, que es un <h1>, y no el componente en sí. Recordemos que React Testing Library se esfuerza por producir un entorno de pruebas que sea lo más parecido a la experiencia del usuario, por lo que es importante que evitemos testear los siguientes casos, pues son detalles de implementación:

- 1. Estado interno de un componente
- 2. Métodos internos de un componente
- 3. Métodos del ciclo de vida de un componente
- 4. Componentes hijos

DigitalHouse>