Testing y pruebas - Jest

Julian F. Latorre

5 de agosto de 2024

Índice

1. Introducción

Este instructivo te guiará a través del proceso de implementación de pruebas unitarias utilizando Jest para un componente React simple. Seguiremos un enfoque paso a paso, desde la creación del componente hasta la escritura y ejecución de las pruebas.

2. Requisitos Previos

Antes de comenzar, asegúrate de tener lo siguiente instalado:

- Node.js y npm
- Create React App (CRA) o un proyecto React existente
- Jest (viene incluido con CRA)

- React Testing Library (viene incluido con CRA)

3. Paso 1: Crear un Componente React

Primero, crearemos un componente React simple para probar. Llamaremos a este componente `Counter`.

Crea un archivo llamado `Counter.js` en tu directorio `src/components/` y agrega el siguiente código:

```
import React, { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);

  const increment = () => setCount(count + 1);
  const decrement = () => setCount(count - 1);

  return (
    <div>
      <h2>Counter: {count}</h2>
      <button onClick={increment}>Increment</button>
      <button onClick={decrement}>Decrement</button>
    </div>
  );
}

export default Counter;
```

4. Paso 2: Crear un Archivo de Prueba

Ahora, crearemos un archivo de prueba para nuestro componente `Counter`. En el mismo directorio, crea un archivo llamado `Counter.test.js`.

5. Paso 3: Importar las Dependencias Necesarias

En `Counter.test.js`, importa las dependencias necesarias:

```
import React from 'react';
import { render, screen, fireEvent } from '@testing-library/react';
import Counter from './Counter';
```

6. Paso 4: Escribir las Pruebas

Ahora, escribiremos algunas pruebas para nuestro componente `Counter`. Agregaremos estas pruebas a `Counter.test.js`:

```
describe('Counter component', () => {
  test('renders initial count of 0', () => {
    render(<Counter />);
```

```
    const countElement = screen.getByText(/Counter: 0/i);
    expect(countElement).toBeInTheDocument();
  });

  test('increments count when increment button is clicked', () => {
    render(<Counter />);
    const incrementButton = screen.getByText(/Increment/i);
    fireEvent.click(incrementButton);
    const countElement = screen.getByText(/Counter: 1/i);
    expect(countElement).toBeInTheDocument();
  });

  test('decrements count when decrement button is clicked', () => {
    render(<Counter />);
    const decrementButton = screen.getByText(/Decrement/i);
    fireEvent.click(decrementButton);
    const countElement = screen.getByText(/Counter: -1/i);
    expect(countElement).toBeInTheDocument();
  });
});
```

7. Paso 5: Ejecutar las Pruebas

Para ejecutar las pruebas, usa el siguiente comando en tu terminal:

```
npm test
```

Este comando iniciará Jest en modo de observación, lo que significa que Jest volverá a ejecutar las pruebas cada vez que guardes cambios en tus archivos.

8. Paso 6: Interpretar los Resultados

Después de ejecutar las pruebas, verás los resultados en la terminal. Si todas las pruebas pasan, verás algo como esto:

```
 PASS  src/components/Counter.test.js
  Counter component
     renders initial count of 0 (23 ms)
     increments count when increment button is clicked (34 ms)
     decrements count when decrement button is clicked (24 ms)

Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        2.145 s
Ran all test suites related to changed files.
```

Si alguna prueba falla, Jest te proporcionará información detallada sobre qué prueba falló y por qué.

9. Resultados Esperados

- El componente se renderiza inicialmente con el contador en 0.
- El contador se incrementa correctamente cuando se hace clic en el botón de incremento.
- El contador se decrementa correctamente cuando se hace clic en el botón de decremento.

Recuerda que este es solo un ejemplo básico. A medida que tus componentes se vuelvan más complejos, puedes expandir tus pruebas para cubrir más casos de uso y escenarios.