Jest

DigitalHouse >
Coding School

Índice

1 | ¿Qué es jest?

Jest es un framework de testing pruebas para JavaScript que nos permite acceder al DOM a través de jsdom, una biblioteca que imita de forma aproximada cómo funciona el navegador. Jest se complementa muy bien con otras herramientas como Enzyme o Testing Library, con la cual trabajaremos más adelante.

2 ¿Cómo funciona Jest?

Jest detecta automáticamente los archivos que sean de tipo .spec o .test y trabaja con ellos. También los que estén guardados dentro de una carpeta llamada __tests__ dentro de src que sean del tipo .js. Veamos un ejemplo sencillo para entender cómo se ve un archivo de test.

Ejemplo práctico I

Supongamos que necesitamos testear esta función. Lo primero que deberíamos hacer es crear el archivo de test ya sea de tipo spec o test (o bien generar la carpeta __tests__). Luego exportamos la función.

```
function suma (a, b) {
    return a + b;
}

module.exports = suma;
```

Ejemplo práctico II

Importamos la función y generamos un test. Para esto añadimos un bloque test el cual recibe dos parámetros: primero un string para darle un título al test y luego una arrow function donde desarrollaremos el código. Utilizaremos la función expect para decidir cuál será la condición con la que el test pasará.

```
{}
import suma from './suma';

test('sumar 2 números', () => {
  expect(suma(1, 2)).toBe(3);
});
```

Ejemplo práctico III

Una vez terminemos de armar nuestros tests, correremos npm test en la terminal y Jest correrá todos los test que encuentre. Se activará un watch mode que correrá los tests automáticamente cada vez que hagamos un cambio. Este comando activa un CLI interactivo que trae varias características para trabajar con los tests, pueden ir probandolas.

Matchers

En el ejemplo de recién vimos que implementamos un matcher, el toBe. Los matchers sirven para ayudarnos a completar el expect, es decir, la condición que debe cumplirse (o no) para que el test pase. Decimos o no porque podríamos haber puesto algo asi: expect(suma(1, 2)).not.toBe(4); Escribir esto es redundante pero sirve de ejemplo para entender que este test pasaría porque 1 + 2 no es 4 por ende se cumple la condición y el test pasa.

Continuamos con..

1. Comparadores
2. Testing de código asíncrono
3. Preparación y desmontaje
4. Test del snapshot

Les recomendamos que continúen leyendo los archivos PDF sobre estos temas para completar la teoría y poder pasar a la práctica.


DigitalHouse>
Coding School