Testing con Javascript PRO



Contenido

Conceptos que vamos a cubrir

- **Tunning Jest**
- Hooks
- Manejo de Excepciones
- Testear promesas
- Mock
- Coverage



Tunning Jest

Tunning Jest

Jest dispone de un buen montón de matchers, pero a veces para la lectura y testeo rápido de unit testing, iría bien poder expandir las capacidades de Jest. Esto lo conseguimos tocando la configuración.

Ejemplos:

- axios-mock-adapter
- jest-extended



Hooks

Hooks

Podemos customizar acciones que se realicen a cada test, antes o después y por cada test suite también.

Estos son:

- beforeAll
- beforeEach
- afterAll
- afterEach

Manejo de excepciones

Manejo de excepciones

En nuestro código en muchas ocasiones tenemos que gestionar excepciones.

Jest nos permite evaluar los tests cuyas funciones o métodos que estamos evaluando lanzan excepciones.

La particularidad es que lo que le vamos a pasar a expect() será un callback y no un valor calculado.

Testear promesas

Testear promesas

Habitualmente nuestro código necesita datos externos, como llamadas a apis de terceros, que se devuelven promesas.

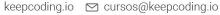
Dado que nosotros estamos programando usando esas apis, lo que deberemos hacer es comprobar que **nuestro código** funciona (pasa los tests) correctamente usando esos recursos de terceros.

No debemos testear los recursos en sí

Testear promesas

Jest nos provee un par de soluciones:

- Invocar aquellas funciones que contengan promesas y una vez se resuelven, testear su correcta manipulación, transformación, etc.
- Utilizar los métodos .resolves y .rejects para evaluar cuando la promise resuelve o rechaza, respectivamente.



- Se conoce a Mock como a los objetos que imitan el comportamiento de objetos reales de una forma controlada.
- Se usan para probar otros objetos en tests unitarios que esperan ciertas respuestas de alguna librería, base de datos o de una clase y esas respuestas no son necesarias para la ejecución de nuestra prueba.
- Ejemplos:
 - Devolver registros de una DB
 - Insertar elementos en una DB
 - Llamadas a apis de terceros que consumen por llamada
 - Imitar registros de actividad en un blog.

- Cada framework de test implementa sus mocks de una forma. En jest podemos crear mocks de cualquier cosa.
- Podemos crear un mock de una clase, una dependencia externa que se encuentre en el node_modules, etc...
- Para crear mocks de una clase en javascript bastaría llamar al método mock de Jest

 En cambio, si usamos Mocha no hay una manera directa de crear mocks, sino que deberíamos apoyarnos en librerías externas como Sinon.js

Gracias al potencial de los mocks, podemos tener métricas de:

- Las veces que se llama una función.
- Los parámetros con los que se ha llamado a dicha función.
- El output que haya generado la llamada al mock

Y también podremos modificar su comportamiento.

En jest, usaremos normalmente:

- .mock: para cargar nuestros propios mocks.
- .fn: para generar funciones mock desde 0.
- .spyOn: para generar también funciones mock de una función ya
- existente.



Coverage

Coverage - Jest

- El coverage es una medida de calidad de nuestras pruebas unitarias.
- Gracias a esto se pueden sacar varias conclusiones:
 - Podemos necesitar más tests
 - Hay código en la app que, actualmente, no se usa y por lo tanto se puede eliminar
- ¿Entonces para testear bien una aplicación hay que tener el coverage al 100%?
- ¿Haría falta testear una función que recibe una string y evalúa esa string con un switch?¿Habría que testear cada una de la salidas posibles de esa función?

Coverage - Jest

- En Jest podemos ver nuestro coverage ejecutando nuestros tests con el flag --coverage, esto nos devuelve una tabla en terminal donde se especifica el porcentaje de código que tenemos testeado en cada uno de los archivos de nuestra app.
- También, en la configuración podemos especificar un mínimo para que hasta que nuestros tests no superen ese porcentaje no sea dado por válido.
- Jest también nos devuelve en coverage en forma de fichero html, mostrando más información acerca del coverage de nuestros tests.

Coverage - Mocha

En Mocha, como para la comprobación de los tests necesitamos una librería externa llamada Istanbul

https://istanbul.js.org/



Madrid | Barcelona | Bogotá

Datos de contacto

https://www.linkedin.com/in/nauelg/

https://twitter.com/gnauel