Curso de Unit Testing con Jest en React

# ¿Qué es un Test? ¿Qué tipos de Test Existen? Jest

## ¿Qué es un test?

Como developers tenemos que garantizar que el codigo escrito cumpla con ciertos requisitos/expectativas. Esto lo hacemos por medio de una prueba (test).

Esto nos asegura:

* Nuestro codigo cumple con el standar.
* Enviamos a producción sin errores.

## Tipos de pruebas

Debemos tener en cuenta que existen dos tipos de pruebas:

* Funcionales.
  + Pruebas Unitarias.- Se prueban pequeñas partes de nuestro codigo asegurandonos así que cumplen con lo que se desea. (En una pagina web las pruebas se traducen a probar cada sección de la página y todas las interacciones en ellas).
* No funcionales.

## Jest

[Jest](https://jestjs.io/) es un framework de JavaScript para pruebas.

Coverage: Esta herramienta nos permite que tanto estamos probando de nuestro proyecto y generar un reporte que nosotros podamos analizar a detalle y ver que nos falta por probar en nuestros proyectos.

**Pruebas funcionales:**  
Son pruebas que se realizan para testear que la funcionalidad del producto sea correcta, es decir, pasarle datos de entrada y recibir los de salida esperados. Hay varios tipos, están las pruebas unitarias, smoke test, sanity, etc.

**Pruebas no funcionales:**  
Con estas testeamos todo lo que no influye a la funcionalidad del producto, como por ejemplo, la accesibilidad, configuración, de rendimiento, entre otras.

# Preparación del entorno con Jest

Escribimos en la terminal el siguiente comando:

npm install -D jest

-D es lo mismo que --save-dev.

Ambos flags guardan la dependencia como una de desarrollo.

## Trabajando con los basicos de Jest

Vamos a estar trabajando con la función “test”, esta recibe dos parametros:

* Un string que describe lo que va a pasar.
* Función anonima en la cual viene lo que se va a probar.

## La función expect

Contiene dos parametros:

* Valor de entrada.
* Con lo que el primer parametro va a ser comparado.

**const** text = "I´m Ironman";

test("It should has the word Ironman", () => {

expect(text).toMatch(/Ironman/);

});

Código listo para hacer test

Ahora para hacer test de nuestro código tenemos que correr el comando test, para eso se tiene que armar en nuestro package.json

"scripts": {

"test": "jest"

},

## Output

PASS src/\*\*test\*\*/global.test.js

✓ It should has the word Ironman (4ms)

Test Suites: 1 passed, 1 total

Tests: 1 passed, 1 total

Snapshots: 0 total

Time: 1.663s

Ran all test suites.

# Implementando pruebas para Boolean y Array



Usamos pruebas unitarias para mejorar la calidad del código en nuestros desarrollos.  
Es necesario ya que nos permite ver cosas que no vemos en el momento en el que se desarrolla.  
Es bueno usar typescript sin embargo también podemos aplicarle pruebas para cubrir cosas que no vemos al desarrollar una funcionalidad.

Por ejemplo:

En este video podemos conocer como probar un callback, números, booleanos y arreglos también.

Me hice la misma pregunta cuando comencé a trabajar formalmente, aunque utilizamos otra suit de pruebas, es tedioso después de agregar un feature tener que hacer las pruebas también o viceversa. Con el tiempo me di cuenta que aunque quita tiempo te da muchas ventajas, por ejemplo:

* Después de modificar código o agregar una nueva característica, corres lo test y estos te ayudan a comprobar que no rompiste alguna otra funcionalidad, así que te sientes más tranquilo.
* Por otro lado si tienes algún servicio de integración continua, que por ejemplo corra los tests cada vez que alguien del equipo cree un pull request, y que bloquee el PR y no habilite a mergeo hasta que no pasen los test o cumpla con el test covegare acordado por el equipo, te estarás asegurando de introducir menos bugs.
* Aunque a veces pueden ser una arma de doble filo, cuando no se testea lo que se debe testear, y resulta que por ejemplo ante algún minimo cambio en la estructura del código más no en su lógica de negocio tengas que cambiar todos los tests.
* En definitiva agregar test a tu proyecto suma seguridad en que tu código hace lo que necesitas.