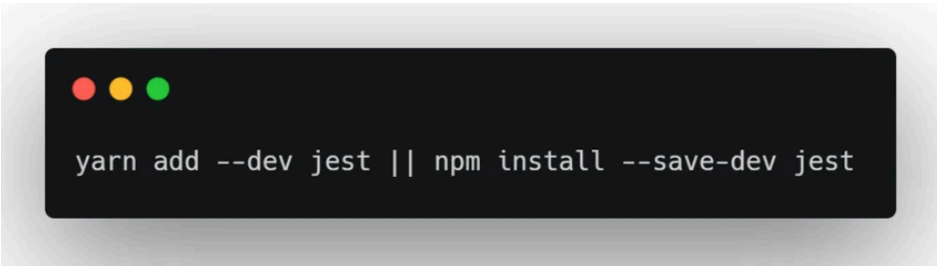


JEST - Guía básica para empezar

01 Instalar


Para configurar tu entorno para trabajar con *Jest*, lo primero que tienes que hacer es instalar la dependencia de desarrollo, dependiendo del manejador de paquetes que estés usando, así:



```
yarn add --dev jest || npm install --save-dev jest
```

02 Scripts

Además, debes añadir en tu “*package.json*”, la configuración en los scripts para decirle que vas a estar usando *Jest* para las pruebas:



```
{
  "scripts": {
    "test": "jest"
  }
}
```

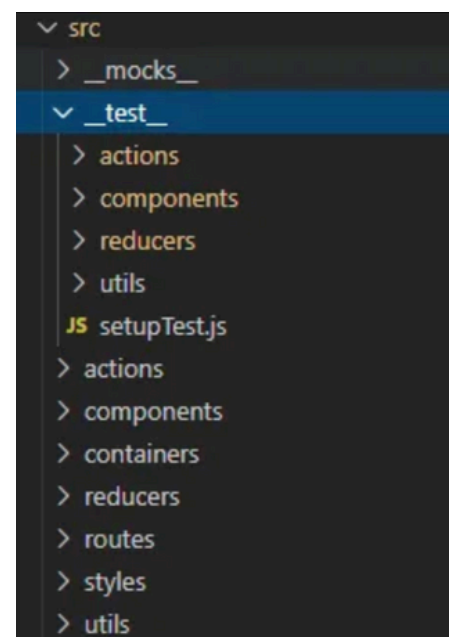
03 Ejecutar

Una vez instalado, los comandos básicos para realizar pruebas son:

- `npm run test`
- `yarn test`

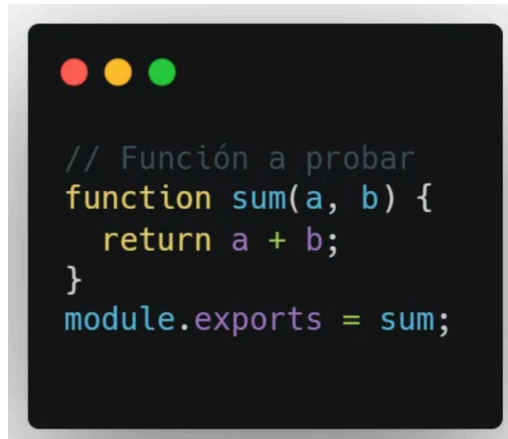
Por convención, lo ideal es que generes una carpeta llamada `_test_`, donde vas a ir creando la estructura de tus pruebas, tal como está organizado tu código.

De esta forma, será más fácil entender a qué corresponde cada prueba que realices.



04 Función Prueba

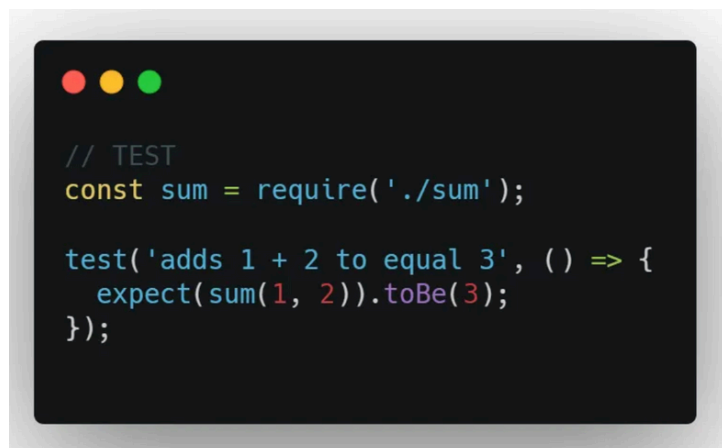
Una de las pruebas más comunes es evaluar si el *output* de una función es equivalente a un resultado específico. Por ejemplo, en una ecuación como $1+1$, siempre te debe dar 2. Imagina que tienes el siguiente archivo “*suma.js*” donde simplemente tienes una función que recibe dos *inputs* y devuelve el resultado de la suma.

A screenshot of a code editor with a dark background and three colored window control buttons (red, yellow, green) at the top left. The code is written in a light-colored font and defines a function named 'sum' that takes two arguments, 'a' and 'b', and returns their sum. The function is then exported as the module's default export.

```
// Función a probar
function sum(a, b) {
  return a + b;
}
module.exports = sum;
```

05 Test Prueba

Si quieres probar que esto siempre va a ejecutarse correctamente, una opción de prueba sería la siguiente. Creas tu archivo *test*, que se llamará exactamente como tu archivo original, pero con una extensión “.*test*”, algo como: “*suma.test.js*”.

A screenshot of a code editor with a dark background and three colored window control buttons (red, yellow, green) at the top left. The code is written in a light-colored font and uses Jest to test the 'sum' function. It imports the 'sum' function from the 'sum' module and writes a test that expects the sum of 1 and 2 to be 3.

```
// TEST
const sum = require('./sum');

test('adds 1 + 2 to equal 3', () => {
  expect(sum(1, 2)).toBe(3);
});
```

06 Métodos comunes de comparación

- **.toBe()** → usa *Object.is* para probar la igualdad exacta.
- **.toEqual()** → comprueba recursivamente cada campo de un *objeto* o *array*.
- **.toBeNull()** → solo coincide con *nulo*.
- **.toBeUndefined()** → solo coincide con *undefined*.
- **.toBeDefined()** → es lo opuesto a *.toBeUndefined()*.
- **.toBeTruthy()** → coincide con cualquier cosa que una declaración *if* trate como verdadera.
- **.toBeFalsy()** → coincide con cualquier cosa que una declaración *if* trate como falsa.

07 Otros métodos de comparación

Comparaciones Numéricas

- `.toBeGreaterThan()` → mayor que.
- `.toBeGreaterThanOrEqual()` → mayor o igual que.
- `.toBeLessThan()` → menor que.
- `.toBeLessThanOrEqual()` → menor o igual que.

Comparaciones Cadenas de Texto

- `.toMatch()` → compara el *string* con una *expresión regular*.
- `.not.toMatch()` → compara el *string* con una *expresión regular* que NO debe estar en el texto.

Iterables

- `.toContain()` → chequea si un *array* u *objeto*, contiene un ítem particular.