

Referencia Rápida de Jest

Tests, matchers, mocks, asíncrono y snapshots

Configuración

Instalación

```
npm install --save-dev jest
# package.json: "scripts": { "test": "jest" }
npx jest # run all tests
npx jest --watch # re-run on changes
```

Nombres de archivos

- *.test.js** Archivos de test (patrón por defecto)
- *.spec.js** Patrón alternativo de test
- __tests__/_** Directorio de tests (descubierto automáticamente)

Ejecutar tests específicos

```
npx jest path/to/file.test.js
npx jest --testNamePattern="adds"
npx jest --verbose # detailed output
```

Tests básicos

Estructura de un test

```
describe("Calculator", () => {
  test("adds 1 + 2 to equal 3", () => {
    expect(add(1, 2)).toBe(3);
  });
});
```

test vs it

```
test("works correctly", () => { /* ... */ });
it("should work correctly", () => { /* ... */ });
// Both are identical; "it" reads like English
```

Omitir y enfocar

- test.skip()** Omitir este test
- test.only()** Ejecutar solo este test
- describe.skip()** Omitir toda la suite
- describe.only()** Ejecutar solo esta suite

Matchers

Igualdad

- .toBe(val)** Igualdad estricta (===)
- .toEqual(val)** Igualdad profunda (objetos/arrays)
- .toStrictEqual(val)** Profunda + tipo + props undefined
- .not.toBe(val)** Negar cualquier matcher

Veracidad

- .toBeTruthy()** Valor verdadero
- .toBeFalsy()** Valor falso
- .toBeNull()** Exactamente **null**
- .toBeUndefined()** Exactamente **undefined**
- .toBeDefined()** No es **undefined**

Números

- .toBeGreaterThan(n)** Mayor que n
- .toBeLessThanOrEqual(n)** Menor o igual que
- .toBeCloseTo(0.3, 5)** Comparación de flotantes (5 dígitos)

Cadenas, arrays y objetos

- .toMatch(/regex/)** La cadena coincide con la regex
- .toContain(item)** El array/iterable contiene el elemento
- .toHaveLength(n)** Longitud del array/cadena
- .toHaveProperty(key, val)** El objeto tiene la propiedad
- .toMatchObject(obj)** El objeto contiene el subconjunto

Tests asíncronos

async / await

```
test("fetches data", async () => {
  const data = await fetchData();
  expect(data).toEqual({ id: 1 });
});
```

Promesas

```
test("resolves to data", () => {
  return expect(fetchData())
    .resolves.toEqual({ id: 1 });
});
```

Rechazos

```
test("rejects with error", async () => {
  await expect(fetchBad())
    .rejects.toThrow("Not Found");
});
```

Excepciones

```
test("throws on invalid input", () => {
  expect(() => validate(null)).toThrow();
  expect(() => validate(null)).toThrow("invalid");
});
```

Mocking

Funciones mock

```
const fn = jest.fn();
fn("hello");
expect(fn).toHaveBeenCalledWith("hello");
expect(fn).toHaveBeenCalledTimes(1);
```

Valores de retorno mock

```
const fn = jest.fn()
  .mockReturnValue(42)
  .mockReturnValueOnce(99);
fn(); // 99 (first call)
fn(); // 42 (subsequent)
```

Módulos mock

```
jest.mock("./api");
const { fetchUser } = require("./api");
fetchUser.mockResolvedValue({ name: "Alice" });
```

Matchers de mock

- .toHaveBeenCalled()** Llamado al menos una vez
- .toHaveBeenCalledTimes(n)** Llamado exactamente n veces
- .toHaveBeenCalledWith(args)** Llamado con argumentos específicos
- .toHaveBeenLastCalledWith(args)** La última llamada tuvo estos argumentos

Espías

Espiarse métodos

```
const spy = jest.spyOn(Math, "random")
  .mockReturnValue(0.5);
expect(Math.random()).toBe(0.5);
spy.mockRestore(); // restore original
```

Espiarse métodos de objetos

```
const obj = { greet: (n) => `Hi ${n}` };
const spy = jest.spyOn(obj, "greet");
obj.greet("Alice");
expect(spy).toHaveBeenCalledWith("Alice");
```

Snapshots

Tests de snapshot

```
test("renders correctly", () => {
  const tree = renderer.create(<App />).toJSON();
  expect(tree).toMatchSnapshot();
});
```

Snapshots en línea

```
test("formats name", () => {
  expect(formatName("alice"))
    .toMatchInlineSnapshot(`"Alice"`);
});
```

Actualizar snapshots

```
npx jest --updateSnapshot # update all
npx jest --updateSnapshot --testNamePattern="renders"
```

Configuración y limpieza

Hooks de ciclo de vida

```
beforeAll(() => { /* once before all tests */ });
afterAll(() => { /* once after all tests */ });
beforeEach(() => { /* before each test */ });
afterEach(() => { /* after each test */ });
```

Alcance

```
describe("Database", () => {
  beforeEach(() => db.connect());
  afterEach(() => db.disconnect());
  test("reads data", () => { /* ... */ });
});
```

Los hooks dentro de describe solo aplican a ese bloque

Configuración

jest.config.js

```
module.exports = {
  testEnvironment: "node",
  coverageThreshold: {
    global: { branches: 80, lines: 80 }
  },
};
```

Opciones comunes

- testEnvironment** "node" o "jsdom" (DOM)
- roots** Directorios donde buscar tests
- collectCoverage** Activar informe de cobertura
- coverageDirectory** Directorio de salida para la cobertura
- moduleNameMapper** Alias de rutas (p. ej., prefijo @/)
- transform** Transformaciones de archivos (Babel, TS, etc.)
- setupFilesAfterFramework** Ejecutar configuración antes de cada suite

Cobertura

```
npx jest --coverage
npx jest --collectCoverageFrom="src/**/*.js"
```

Referencia Rápida de Jest

Patrones comunes

Testear llamadas a API

```
jest.mock("./api");
test("loads users", async () => {
  api.getUsers.mockResolvedValue([{id: 1}]);
  const users = await loadUsers();
  expect(users).toHaveLength(1);
});
```

Mocks de temporizadores

```
jest.useFakeTimers();
test("delays execution", () => {
  const cb = jest.fn();
  setTimeout(cb, 1000);
  jest.advanceTimersByTime(1000);
  expect(cb).toHaveBeenCalled();
});
```

Tests parametrizados

```
test.each([
  [1, 1, 2],
  [2, 3, 5],
])("add(%i, %i) = %i", (a, b, expected) => {
  expect(add(a, b)).toBe(expected);
});
```

Matchers personalizados

```
expect.extend({
  toBeWithinRange(received, floor, ceil) {
    const pass = received >= floor
      && received <= ceil;
    return { pass, message: () =>
      `expected ${received} in [${floor},${ceil}]` };
  }
});
```