

# Jest Riferimento Rapido

Test, matcher, mock, async e snapshot

## Configurazione

### Installazione

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

### Nomenclatura dei File

**\*.test.js** File di test (pattern predefinito)  
**\*.spec.js** Pattern di test alternativo  
**\_\_tests\_\_/\_** Directory di test (auto-scoperta)

### Esecuzione di Test Specifici

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

## Test di Base

### Struttura del 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
```

### Salta e Focalizza

**test.skip()** Salta questo test  
**test.only()** Esegui solo questo test  
**describe.skip()** Salta l'intera suite  
**describe.only()** Esegui solo questa suite

## Matcher

### Uguaglianza

**.toBe(val)** Uguaglianza stretta (===)  
**.toEqual(val)** Uguaglianza profonda (oggetti/array)  
**.toStrictEqual(val)** Profonda + tipo + prop undefined  
**.not.toBe(val)** Nega qualsiasi matcher

### Veridicità

**.toBeTruthy()** Valore truthy  
**.toBeFalsy()** Valore falsy  
**.toBeNull()** Esattamente **null**  
**.toBeUndefined()** Esattamente **undefined**  
**.toBeDefined()** Non **undefined**

### Numeri

**.toBeGreaterThan(n)** Maggiore di n  
**.toBeLessThanOrEqual(n)** Minore o uguale  
**.toBeCloseTo(0.3, 5)** Confronto float (5 cifre)

### Stringhe, Array, Oggetti

**.toMatch(/regex/)** La stringa corrisponde alla regex  
**.toContain(item)** Array/iterabile contiene l'elemento  
**.toHaveLength(n)** Lunghezza dell'array/stringa  
**.toHaveProperty(key, val)** L'oggetto ha la proprietà  
**.toMatchObject(obj)** L'oggetto contiene il sottoinsieme

## Test Asincroni

### async / await

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

### Promise

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

### Rifiuti

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

### Eccezioni

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

## Mock

### Funzioni Mock

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

### Valori di Ritorno Mock

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

### Moduli Mock

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

### Matcher per Mock

**.toHaveBeenCalled()** Chiamato almeno una volta  
**.toHaveBeenCalledTimes(n)** Chiamato esattamente n volte  
**.toHaveBeenCalledWith(args)** Chiamato con argomenti specifici  
**.toHaveBeenLastCalledWith(args)** L'ultima chiamata aveva questi argomenti

## Spy

### Spiare i Metodi

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

## Spiare i Metodi di Oggetti

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

## Snapshot

### Test Snapshot

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

### Snapshot Inline

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

### Aggiornamento Snapshot

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

## Setup e Teardown

### Hook del Ciclo di Vita

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

### Scoping

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

Gli hook dentro describe si applicano solo a quel blocco

## Configurazione

### jest.config.js

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

### Opzioni Comuni

**testEnvironment** "node" o "jsdom" (DOM)  
**roots** Directory in cui cercare i test  
**collectCoverage** Abilita il report di copertura  
**coverageDirectory** Directory di output per la copertura  
**moduleNameMapper** Alias percorsi (es. prefisso @/)  
**transform** Trasformazioni file (Babel, TS, ecc.)  
**setupFilesAfterFramework** Esegui setup prima di ogni suite

### Copertura

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

# Jest Riferimento Rapido

---

## Pattern Comuni

### Test delle Chiamate API

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

### Mock dei Timer

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

### Test Parametrizzati

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

### Matcher Personalizzati

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