Referência Rápida de Jest

Testes, matchers, mocks, assíncrono e snapshots

Configuração

Instalação

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

Nomenclatura de Arquivos

`*.test.js`	Arquivos de teste (padrão)
`*.spec.js`	Padrão alternativo de teste
`__tests__/`	Diretório de testes (descoberto automaticamente)

Executando Testes Específicos

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

Testes Básicos

Estrutura do Teste

```
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
```

Pular e Focar

`test.skip()`	Pula este teste
`test.only()`	Executa apenas este teste
`describe.skip()`	Pula toda a suíte
`describe.only()`	Executa apenas esta suíte

Matchers

Igualdade

`.toBe(val)`	Igualdade estrita (`===`)
`.toEqual(val)`	Igualdade profunda (objetos/arrays)
`.toStrictEqual(val)`	Profundo + tipo + props undefined
`.not.toBe(val)`	Nega qualquer matcher

Veracidade

`.toBeTruthy()`	Valor verdadeiro
`.toBeFalsy()`	Valor falso
`.toBeNull()`	Exatamente **null**
`.toBeUndefined()`	Exatamente **undefined**
`.toBeDefined()`	Diferente de **undefined**

Números

`.toBeGreaterThan(n)`	Maior que n
`.toBeLessThanOrEqual(n)`	Menor ou igual
`.toBeCloseTo(0.3, 5)`	Comparação de ponto flutuante (5 dígitos)

Strings, Arrays, Objetos

`.toMatch(/regex/)`	String corresponde ao regex
`.toContain(item)`	Array/iterável contém o item
`.toHaveLength(n)`	Comprimento do array/string
`.toHaveProperty(key, val)`	Objeto possui propriedade
`.toMatchObject(obj)`	Objeto contém subconjunto

Testes Assíncronos

async / await

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

Promises

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

Rejeições

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

Exceções

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

Mocking

Funções 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)
```

Mockando Módulos

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

Matchers de Mock

`.toHaveBeenCalled()`	Chamado ao menos uma vez
`.toHaveBeenCalledTimes(n)`	Chamado exatamente n vezes
`.toHaveBeenCalledWith(args)`	Chamado com argumentos específicos
`.toHaveBeenLastCalledWith(args)`	Última chamada teve esses argumentos

Spies

Espionando Métodos

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

Espionando Métodos de Objeto

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

Snapshots

Teste de Snapshot

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

Snapshots Inline

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

Atualizando Snapshots

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

Setup e Teardown

Hooks de Ciclo de Vida

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

Escopo

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

Hooks dentro de describe se aplicam apenas a esse bloco

Configuração

jest.config.js

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

Opções Comuns

`testEnvironment`	`"node"` ou `"jsdom"` (DOM)
`roots`	Diretórios para buscar testes
`collectCoverage`	Habilita relatório de cobertura
`coverageDirectory`	Diretório de saída para cobertura
`moduleNameMapper`	Aliases de caminho (ex.: prefixo `@/`)
`transform`	Transformações de arquivo (Babel, TS, etc.)
`setupFilesAfterFramework`	Executa setup antes de cada suíte

Cobertura

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

Referência Rápida de Jest

Padrões Comuns

Testando Chamadas de API

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

Mocks de Timer

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

Testes 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}]` };
}});
```