JEST QUICK REFERENCE

Tests, matchers, mocking, async, and snapshots

Setup

Installation

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

File Naming

`*.test.js`	Test files (default pattern)
`*.spec.js`	Alternative test pattern
`__tests__/`	Test directory (auto-discovered)

Running Specific Tests

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

Basic Tests

Test Structure

```
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
```

Skipping & Focusing

`test.skip()`	Skip this test
`test.only()`	Run only this test
`describe.skip()`	Skip entire suite
`describe.only()`	Run only this suite

Matchers

Equality

`.toBe(val)`	Strict equality (`===`)
`.toEqual(val)`	Deep equality (objects/arrays)
`.toStrictEqual(val)`	Deep + type + undefined props
`.not.toBe(val)`	Negate any matcher

Truthiness

`.toBeTruthy()`	Truthy value
`.toBeFalsy()`	Falsy value
`.toBeNull()`	Exactly `null`
`.toBeUndefined()`	Exactly `undefined`
`.toBeDefined()`	Not `undefined`

Numbers

`.toBeGreaterThan(n)`	Greater than n
`.toBeLessThanOrEqual(n)`	Less than or equal
`.toBeCloseTo(0.3, 5)`	Float comparison (5 digits)

Strings, Arrays, Objects

`.toMatch(/regex/)`	String matches regex
`.toContain(item)`	Array/iterable contains item
`.toHaveLength(n)`	Length of array/string
`.toHaveProperty(key, val)`	Object has property
`.toMatchObject(obj)`	Object contains subset

Async Testing

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 });
});
```

Rejections

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

Exceptions

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

Mocking

Mock Functions

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

Mock Return Values

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

Mocking Modules

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

Mock Matchers

`.toHaveBeenCalled()`	Called at least once
`.toHaveBeenCalledTimes(n)`	Called exactly n times
`.toHaveBeenCalledWith(args)`	Called with specific arguments
`.toHaveBeenLastCalledWith(args)`	Last call had these arguments

Spies

Spying on Methods

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

Spying on Object Methods

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

Snapshots

Snapshot Testing

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

Inline Snapshots

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

Updating Snapshots

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

Setup & Teardown

Lifecycle Hooks

```
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", () => { /* ... */ });
});
```

Hooks inside describe only apply to that block

Configuration

jest.config.js

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

Common Options

`testEnvironment`	`"node"` or `"jsdom"` (DOM)
`roots`	Directories to search for tests
`collectCoverage`	Enable coverage reporting
`coverageDirectory`	Output directory for coverage
`moduleNameMapper`	Path aliases (e.g., `@/` prefix)
`transform`	File transforms (Babel, TS, etc.)
`setupFilesAfterFramework`	Run setup before each suite

Coverage

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

Common Patterns

Testing API Calls

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

Timer Mocks

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

Parameterized Tests

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

Custom Matchers

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