Jest

Tests, matchers and mock functions in Jest 30.

01 Install & run

```
npm install --save-dev jest
# TypeScript: types and a transformer
npm install --save-dev @types/jest ts-jest
# package.json scripts: "test": "jest"
npm test
```

Jest 30 needs Node 18.14, 20, 22 or 24+, and its types TypeScript 5.4+. Test files are *.test.js or *.spec.js (also ts, tsx, mjs, cjs) and anything in a __tests__ folder.

02 Test structure

```
describe('Calculator', () => {
  test('adds 1 + 2 to equal 3', () => {
    expect(1 + 2).toBe(3)
  })
  it('subtracts 5 - 2 to equal 3', () => {
    expect(5 - 2).toBe(3)
  })
})
```

it is an alias of test. test.only runs just that test in its file; test.skip skips it and test.todo('name') lists one to write.

03 Setup & teardown

```
let service
beforeAll(() => database.connect())
beforeEach(() => {
  service = new UserService()
})
afterEach(() => service.cleanup())
afterAll(() => database.disconnect())
```

The All hooks run once, the Each hooks around every test in their describe. Jest waits for a returned promise.

04 Equality

```
const a = { id: 1, tag: undefined }
expect(a.id).toBe(1)           // Object.is
expect(a).not.toBe({ id: 1 })  // not same
expect(a).toEqual({ id: 1 })   // deep
expect(a).not.toStrictEqual({ id: 1 })
expect(0.1 + 0.2).not.toBe(0.3)
expect(0.1 + 0.2).toBeCloseTo(0.3)
```

toEqual ignores undefined properties; toStrictEqual counts them and checks class types. Floats: toBeCloseTo.

05 Common matchers

`.not.toBe(x)`	Negates any matcher
`toBeTruthy()`	Also toBeFalsy(), toBeNull()
`toBeGreaterThan(3)`	Also toBeLessThanOrEqual(5)
`toMatch(/stop/)`	Regex or substring
`toContain(item)`	In an array, Set or string
`toHaveProperty(key)`	Key path, optional value
`toMatchObject(obj)`	Subset of an object matches

06 Errors & asymmetric matchers

```
const fail = () => { throw new Error('No') }
expect(fail).toThrow('No')   // substring
expect(fail).toThrow(Error)  // class
expect({ id: 7, tags: ['a'] }).toEqual({
  id: expect.any(Number),
  tags: expect.arrayOf(expect.any(String)),
})
```

toThrow needs a function: expect(fail()) throws before Jest can catch it. expect.arrayOf is new in Jest 30.

07 Mock functions

```
const fn = jest.fn()
fn('a', 1)                    // undefined
const double = jest.fn(x => x * 2)
double(5)                     // 10
const roll = jest.fn().mockReturnValue(1)
roll.mockReturnValueOnce(6)
roll()                        // 6
roll()                        // 1
```

mockImplementation(fn) swaps the body. Async mocks: mockResolvedValue(v) and mockRejectedValue(err).

08 Calls & results

```
expect(fn).toHaveBeenCalledTimes(1)
expect(fn).toHaveBeenCalledWith('a', 1)
expect(fn).toHaveBeenLastCalledWith('a', 1)
expect(double).toHaveReturnedWith(10)
fn.mock.calls                 // [['a', 1]]
double.mock.results[0].value  // 10
```

Jest 30 removed the aliases (toBeCalled, toBeCalledWith, toReturn, toThrowError and the rest): use these full names.

09 Spies & cleanup

```
const spy = jest.spyOn(video, 'play')
video.play()          // the real method runs
expect(spy).toHaveBeenCalled()
spy.mockClear()       // forget the calls
spy.mockReset()       // + no implementation
spy.mockRestore()     // + original is back
```

The clearMocks, resetMocks and restoreMocks config options do this before every test. jest.restoreAllMocks() only restores spies: jest.fn() mocks keep their state.

Jest

Module mocks, async code, timers and config.

10 Module mocks

```
import { fetchUser } from './api'
jest.mock('./api') // exports are jest.fn()
test('loads a profile', async () => {
  fetchUser.mockResolvedValue({ id: 1 })
  const profile = await loadProfile(1)
  expect(profile).toEqual({ id: 1 })
})
```

babel-jest and ts-jest hoist jest.mock above the imports. Native ESM (still experimental) cannot: use jest.unstable_mockModule() and await import(). A mock applies to this file only, and Jest 30 paths are case-sensitive. TypeScript: jest.mocked(fetchUser).

11 Factories & partial mocks

```
jest.mock('./math', () => ({
  ...jest.requireActual('./math'),
  add: jest.fn(() => 5),  // rest stays real
}))
```

The hoisted factory can only use variables named mock*. Default export: return { __esModule: true, default: jest.fn() }. A __mocks__ folder beside the module holds a manual mock.

12 Async tests

```
test('resolves and rejects', async () => {
  expect(await fetchData()).toBe('data')
  await expect(fetchError())
    .rejects.toThrow('error')
})
```

Always await or return the promise, .resolves and .rejects included, or the test ends first. If a catch block holds the only assertion, start with expect.assertions(1).

13 Fake timers

```
beforeEach(() => jest.useFakeTimers())
afterEach(() => jest.useRealTimers())
test('runs after one second', () => {
  const callback = jest.fn()
  setTimeout(callback, 1000)
  jest.advanceTimersByTime(1000)
  expect(callback).toHaveBeenCalledTimes(1)
})
```

runAllTimers() runs everything queued. For setInterval or a timer that reschedules itself, use runOnlyPendingTimers().

14 Dates & async timers

```
jest.setSystemTime(new Date('2024-06-15'))
const promise = delayedOperation()
await jest.runAllTimersAsync()
expect(await promise).toBe('completed')
```

With fake timers on, Date is faked too. If code awaits before it sets a timer, use the Async versions (also advanceTimersByTimeAsync): promise callbacks run first.

15 Custom matchers

```
expect.extend({
  toBeWithinRange(n, min, max) {
    const pass = n >= min && n <= max
    const not = pass ? 'not ' : ''
    const message = () =>
      `expected ${n} ${not}in ${min}-${max}`
    return { pass, message }
  },
})
```

Use: expect(100).toBeWithinRange(90, 110).

16 Snapshots & React components

```
/** @jest-environment jsdom */
test('renders the card', () => {
  const { container } = render(<Card />)
  expect(container).toMatchSnapshot()
})
```

The first run writes a .snap file to commit. Later runs fail on any change; jest -u accepts it. render is from React Testing Library; jsdom needs the jest-environment-jsdom package.

17 jest.config.js & coverage

```
module.exports = {
  preset: 'ts-jest',          // TypeScript
  testEnvironment: 'node',    // or 'jsdom'
  setupFilesAfterEnv: ['./jest.setup.js'],
  coverageThreshold: {
    global: { branches: 80, lines: 80 },
  },
}
```

Coverage under a threshold fails the run. Reports land in coverage/. Exclude code with /* istanbul ignore next */.

18 CLI

jest user	Files with user in the path
-t "adds"	Tests whose name matches
--watch	Rerun tests of changed files
--testPathPatterns	Was --testPathPattern in 29
-u	Update snapshots
--ci	Never write new snapshots

Through npm, flags go after --: npm test -- --coverage.