

# Jest 빠른 참조

테스트, 매치, 목킹, 비동기, 스냅샷

## 설정

### 설치

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

### 파일 네이밍

**\*.test.js** 테스트 파일 (기본 패턴)  
**\*.spec.js** 대체 테스트 패턴  
**\_\_tests\_\_/** 테스트 디렉터리 (자동 탐색)

### 특정 테스트 실행

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

## 기본 테스트

### 테스트 구조

```
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
```

### 테스트 건너뛰기 & 집중

**test.skip()** 이 테스트 건너뛰기  
**test.only()** 이 테스트만 실행  
**describe.skip()** 전체 스위트 건너뛰기  
**describe.only()** 이 스위트만 실행

## 매치

### 동등성

**.toBe(val)** 엄격한 동등성 (===)  
**.toEqual(val)** 깊은 동등성 (객체/배열)  
**.toStrictEqual(val)** 깊은 동등성 + 타입 + undefined 속성  
**.not.toBe(val)** 매치 부정

### 참/거짓

**.toBeTruthy()** truthy 값  
**.toBeFalsy()** falsy 값  
**.toBeNull()** 정확히 null  
**.toBeUndefined()** 정확히 undefined  
**.toBeDefined()** undefined가 아님

### 숫자

**.toBeGreaterThan(n)** n보다 큼  
**.toBeLessThanOrEqual(n)** n보다 작거나 같음  
**.toBeCloseTo(0.3, 5)** 부동소수점 비교 (5자리)

## 문자열, 배열, 객체

**.toMatch(regex/)** 문자열이 정규식과 일치  
**.toContain(item)** 배열/이터러블에 항목 포함  
**.toHaveLength(n)** 배열/문자열 길이  
**.toHaveProperty(key, val)** 객체에 속성 있음  
**.toMatchObject(obj)** 객체가 부분집합 포함

## 비동기 테스트

### 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 });
});
```

### 거부(Rejection)

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

### 예외

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

## 목킹

### 목 함수

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

### 목 반환값

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

### 모듈 목킹

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

### 목 매치

**.toHaveBeenCalled()** 최소 한 번 호출됨  
**.toHaveBeenCalledTimes(n)** 정확히 n번 호출됨  
**.toHaveBeenCalledWith(args)** 특정 인수로 호출됨  
**.toHaveBeenLastCalledWith(args)** 마지막 호출에서 인수 사용

## 스파이

### 메서드 스파이

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

### 객체 메서드 스파이

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

## 스냅샷

### 스냅샷 테스트

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

### 인라인 스냅샷

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

### 스냅샷 업데이트

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

## 설정 및 정리

### 라이프사이클 훅

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

### 스코프

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

describe 내부의 훅은 해당 블록에만 적용됨

## 설정

### jest.config.js

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

# Jest 빠른 참조

## 주요 옵션

|                                 |                         |
|---------------------------------|-------------------------|
| <b>testEnvironment</b>          | "node" 또는 "jsdom" (DOM) |
| <b>roots</b>                    | 테스트를 탐색할 디렉터리           |
| <b>collectCoverage</b>          | 커버리지 리포트 활성화            |
| <b>coverageDirectory</b>        | 커버리지 결과 출력 디렉터리         |
| <b>moduleNameMapper</b>         | 경로 별칭 (예: @/ 접두사)       |
| <b>transform</b>                | 파일 변환기 (Babel, TS 등)    |
| <b>setupFilesAfterFramework</b> | 각 스위트 전에 설정 실행          |

## 커버리지

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

## 주요 패턴

### API 호출 테스트

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

### 타이머 목킹

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

### 매개변수화된 테스트

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

### 커스텀 매치

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