

# JEST クイックリファレンス

テスト、マッチャー、モック、非同期、スナップショット

## セットアップ

### インストール

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

### ファイル命名

- `*.test.js` テストファイル (デフォルトパターン)
- `*.spec.js` 代替テストパターン
- `__tests__/*` テストディレクトリ (自動検出)

### 特定テストの実行

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

## 基本テスト

### テスト構造

```
describe("Calculator", () => {
  test("adds 1 + 2 to equal 3", () => {
    expect(1 + 2).toBe(3);
  });
});
```

### test と it

```
test("works correctly", () => { /* ... */ });
it("should work correctly", () => { /* ... */ });
// Both are identical; "it" reads like English
```

### スキップとフォーカス

- `test.skip()` このテストをスキップ
- `test.only()` このテストのみ実行
- `describe.skip()` スイート全体をスキップ
- `describe.only()` このスイートのみ実行

## マッチャー

### 等値

- `toBe(val)` 厳密等値 (``===``)
- `toEqual(val)` 深い等値 (オブジェクト / 配列)
- `toStrictEqual(val)` 深い等値 + 型 + undefined プロパティ
- `not.toBe(val)` 任意のマッチャーを否定

### 真偽値

- `toBeTruthy()` truthy な値
- `toBeFalsy()` falsy な値
- `toBeNull()` 厳密に `null`
- `toBeUndefined()` 厳密に `undefined`
- `toBeDefined()` `undefined` でない

### 数値

- `toBeGreaterThan(n)` n より大きい
- `toBeLessThanOrEqual(n)` n 以下
- `toBeCloseTo(0.3, 5)` 浮動小数点比較 (5桁精度)

### 文字列・配列・オブジェクト

- `toMatch(regex/)` 文字列が正規表現にマッチ
- `toContain(item)` 配列 / イテラブルが要素を含む
- `toHaveLength(n)` 配列 / 文字列の長さ
- `toHaveProperty(key, val)` オブジェクトがプロパティを持つ
- `toMatchObject(obj)` オブジェクトがサブセットを含む

## 非同期テスト

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

### リジェクション

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

### 例外

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

## モック

### モック関数

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

### モックの戻り値

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

### モジュールのモック

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

### モックマッチャー

- `toHaveBeenCalledWith()` 少なくとも 1 回呼ばれた

- `toHaveBeenCalledWith(n)` ちょうど n 回呼ばれた
- `toHaveBeenCalledWith(args)` 特定の引数で呼ばれた
- `toHaveBeenCalledWith(args)` 最後の呼び出しがこれらの引数だった

## スパイ

### メソッドのスパイ

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

### オブジェクトメソッドへのスパイ

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

## スナップショット

### スナップショットテスト

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

### インラインスナップショット

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

### スナップショットの更新

```
npm test --updateSnapshot # update all
npm test --updateSnapshot --testNamePattern="renders"
```

## セットアップとティアダウン

### ライフサイクルフック

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

### スコープ

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

describe 内のフックはそのブロックにのみ適用される

## 設定

### jest.config.js

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

### よく使うオプション

- `testEnvironment` `node` または `jsdom` (DOM)

- `roots` テストを検索するディレクトリ
- `collectCoverage` カバレッジレポートを有効化
- `coverageDirectory` カバレッジの出力ディレクトリ
- `moduleNameMapper` パスエイリアス (例: @/ プレフィックス)
- `transform` ファイル変換 (Babel, TS など)
- `setupFilesAfterFramework` 各スイート前にセットアップを実行

### カバレッジ

```
npm test --coverage
npm test --collectCoverageFrom="src/**/*.js"
```

### よく使うパターン

#### API 呼び出しのテスト

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

#### タイマーモック

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

#### パラメータ化テスト

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

#### カスタムマッチャー

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