|  |  |  |  |
| --- | --- | --- | --- |
| **Jest Method** | **Description** | **Use Case** | **Practice Tip** |
| **test() / it()** | Defines an individual test | Write test cases with expected behavior | **test('adds 1 + 2 to equal 3', () => { expect(1 + 2).toBe(3); });** |
| **describe()** | Groups related  tests together | Organize test suites for a component or function | Group tests by feature:  **describe('Login Form', () => { ... })** |
| **expect()** | Used to assert values | Base for all matchers | **expect(value).toBe(expected)** |
| **toBe()** | Strict equality (===) | Compare primitives or exact values | **expect(sum(2, 3)).toBe(5)** |
| **toEqual()** | Deep equality (objects, arrays) | Compare objects/arrays | **expect({ a: 1 }).toEqual({ a: 1 })** |
| **toBeTruthy()** | Asserts value is truthy | Boolean checks | **expect(isLoggedIn).toBeTruthy()** |
| **toBeFalsy()** | Asserts value is falsy | Boolean checks | **expect(error).toBeFalsy()** |
| **toContain()** | Checks if array or string contains a value | Array includes or substring check | **expect(['a', 'b']).toContain('a')** |
| **toHaveLength()** | Checks length of array/string | Length checks | **expect(items).toHaveLength(3)** |
| **toBeDefined()** | Checks that a variable is defined | Presence check | **expect(user.name).toBeDefined()** |
| **toBeNull()** | Checks if value is exactly null | Null-check | **expect(response).toBeNull()** |
| **toMatch()** | Regex match on strings | Pattern matching | **expect('abc123').toMatch(/\d+/)** |
| **jest.fn()** | Creates a mock function | Unit test internal calls or props | **const mockFn = jest.fn();** |
| **.toHaveAttribute(‘attribute name)** |  |  |  |
| **jest.spyOn()** | Spy on a method of an object | Monitor calls to real methods | **jest.spyOn(obj, 'method')** |
| **mockReturnValue()** | Force a mock function to return a value | Stub function behavior | **mockFn.mockReturnValue(42)** |
| **mockResolvedValue()** | Force a promise-returning mock to resolve | Stub async API responses | **mockFn.mockResolvedValue({ data: 'ok' })** |
| **mockRejectedValue()** | Force a promise-returning mock to reject | Stub failed async calls | **mockFn.mockRejectedValue(new Error('fail'))** |
| **beforeEach()** | Runs setup code before each test | Reuse setup like rendering or variable reset | **beforeEach(() => setup())** |
| **afterEach()** | Runs cleanup code after each test | Cleanup DOM or mocks | **afterEach(() => cleanup())** |
| **jest.clearAllMocks()** | Resets all mock usage data (calls, instances) | Before each test for isolation | **Place in beforeEach** |

**Practice Ideas**

1. **Utility Function Testing**
   * Write functions like add(), capitalize(), isEven()
   * Use toBe, toEqual, toMatch, toBeTruthy, toBeFalsy
2. **Mock API Calls**
   * Create a fake fetchData() function
   * Use jest.fn() + mockResolvedValue() to simulate success/failure
   * Assert loading/success/error states
3. **Spy on Method**
   * Use jest.spyOn() to track internal method call
   * Example: a logger.log() function that gets called on submit
4. **Array Manipulations**
   * Write a function like filterAdults(users)
   * Test with toHaveLength, toContain, toEqual
5. **Setup/Cleanup**
   * Use beforeEach() and afterEach() to reset state
   * Practice with multiple describe() blocks