Jest

Jest là gì?

Là công cụ được xây dựng bởi Facebook, một công cụ mạnh mẽ hỗ trợ cho việc viết unit test trên các ứng dụng Javascript.

Cài đặt

Có thể dùng command npm hoặc yarn để cài đặt

npm install --save-dev jest

Hoặc

yarn add --dev jest

Thêm vào đoạn script tại package.json

"scripts": {

"test": "jest"

}

Tạo file test với dạng \*.test.js

Demo:

// file DemoSimple.js

const add = (a, b) => a + b;

module.exports = add;

// file .test.js

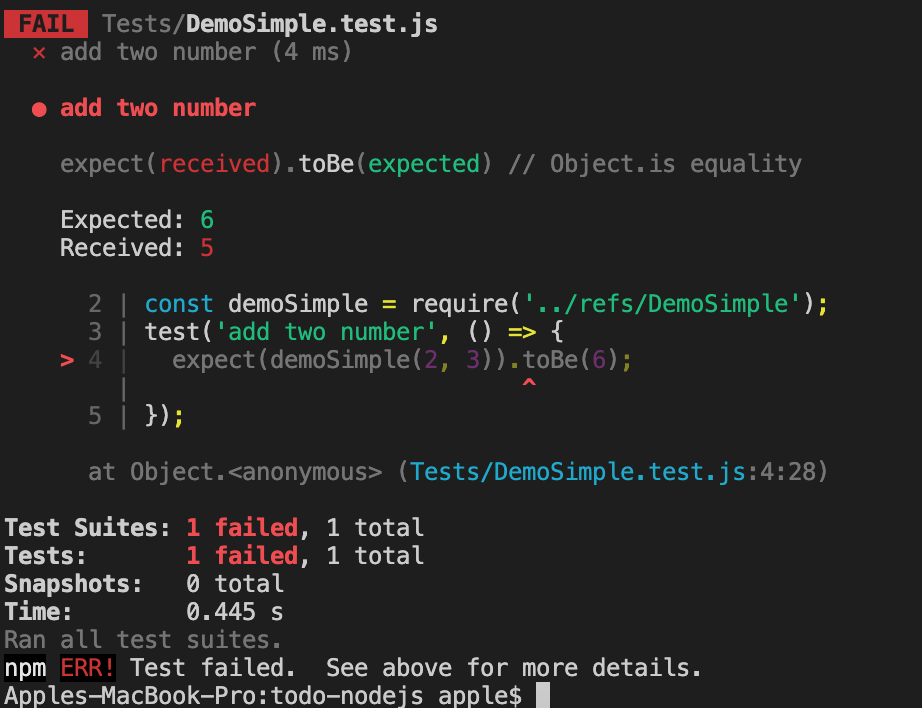
const demoSimple = require('../refs/DemoSimple');

test('add two number', () => {

expect(demoSimple(2, 3)).toBe(6);

});

Chạy lệnh npm test được kết quả



Các toán từ(matchers) trong Jest

toBe

Sử dụng toBe để kiểm tra các toán tử trả về

expect(add(2, 2)).toBe(4);

toEqual

So sánh kết quả trả về dạng object

const add = (a, b) => ({ a, b });

test('check obj', () => {

expect(add(2, 2)).toEqual({ a: 2, b: 2 });

});

## Truthiness

Phân biệt giữa undefined, null và false...

* toBeNull chỉ đối sánh với null
* toBeUndefined chỉ đối sánh không xác định
* toBeDefined đối lập với toBeUndefined
* toBeTruthy khớp với bất kỳ thứ gì mà câu lệnh if coi là true
* toBeFalsy khớp với bất kỳ thứ gì mà câu lệnh if coi là sai

test('null', () => {

const n = null;

expect(n).toBeNull();

expect(n).toBeDefined();

expect(n).not.toBeUndefined();

expect(n).not.toBeTruthy();

expect(n).toBeFalsy();

});

test('zero', () => {

const z = 0;

expect(z).not.toBeNull();

expect(z).toBeDefined();

expect(z).not.toBeUndefined();

expect(z).not.toBeTruthy();

expect(z).toBeFalsy();

});

Number

Có các biểu thức so sánh như sau:

test('test matching number', () => {

const value = 2 + 2;

expect(value).toBeGreaterThan(3);

expect(value).toBeGreaterThanOrEqual(3.5);

expect(value).toBeLessThan(5);

expect(value).toBeLessThanOrEqual(4.5);

// Với dữ liệu dạng số thì các bạn có thể dùng toBe hoặc toEqual

// để so sánh bằng đều được

expect(value).toBe(4);

expect(value).toEqual(4);

});

String

Các biểu thức so sánh string như sau

test('contain I', () => {

expect('team').not.toMatch(/I/);

});

test('Not contain abc', () => {

expect('Sabcdasdas').toMatch(/abc/);

});

Arrays

Sử dụng biểu thức contain

test('contain abc', () => {

expect(['abc', 'xyz']).toContain('abc');

});

function divideByZero() {

throw new Error('can not devide to 0');

}

test('test throw exeption divideByZero', () => {

expect(() => divideByZero()).toThrow();

expect(() => divideByZero()).toThrow(Error);

// compare message or use regex

expect(() => divideByZero()).toThrow('can not devide to 0');

expect(() => divideByZero()).toThrow(/can not/);

});

Async - promise

doublefunc = function(number) {

return new Promise((resolve, reject) => {

if(number%2 == 0) {

resolve(number\*2)

}

else {

reject("error 1")

}

})

}

test('test doublefunc', () => {

return demoSimple.doublefunc(12).then(data => {

expect(data).toBe(24);

});

});

async/await

doubleFunc = function(number) {

return new Promise((resolve, reject) => {

if(number%2 == 0) {

resolve(number\*2)

}

else {

reject("error 1")

}

})

}

syncFunc = async function(data) {

try {

const result = await compare1(data)

return result

} catch (err) {

throw err

}

}

Test

test('test async', async () => {

const data = await demoSimple.syncFunc(12);

expect(data).toBe(24);

});