Relatório de Testes de funções com jest

**Funções de soma simples**

const soma1 = (positivo1 , positivo2) => {

    return positivo1 + positivo2

}

console.log(soma1(2, 3))

const soma2 = (positivo , negativo) => {

    return positivo + negativo

}

console.log(soma1(2, -3))

const soma3 = (negativo1 , negativo2) => {

    return negativo1 + negativo2

}

console.log(soma1(-2, -3))

describe('teste de somas', () => {

    it('deve somar dois positivos', () => {

        const esperado = 5;

        const retornado = soma1(2, 3);

        expect(retornado).toBe(esperado);

    })

    it('deve somar um positivo e um negativo', () => {

        const esperado = -1;

        const retornado = soma2(2, -3);

        expect(retornado).toBe(esperado);

    })

    it('deve somar dois negativos', () => {

        const esperado = -5;

        const retornado = soma3(-2, -3);

        expect(retornado).toBe(esperado);

    })

})

* Nos três testes os valores retornados se mostraram ser o mesmo do esperado, respectivamente: 5, -1 e -5

**Funções de subtração**

const subtracao1 = (positivo1 , positivo2) => {

    return positivo1 - positivo2

}

console.log(subtracao1(2, 3))

const subtracao2 = (positivo , negativo) => {

    return positivo - negativo

}

console.log(subtracao2(2, -3))

const subtracao3 = (negativo1 , negativo2) => {

    return negativo1 - negativo2

}

console.log(subtracao3(-2, -3))

describe('teste de subtrações', () => {

    it('deve subtrair dois positivos', () => {

        const esperado = -1;

        const retornado = subtracao1(2, 3);

        expect(retornado).toBe(esperado);

    })

    it('deve subtrair um positivo e um negativo', () => {

        const esperado = 5;

        const retornado = subtracao2(2, -3);

        expect(retornado).toBe(esperado);

    })

    it('deve subtrair dois negativos', () => {

        const esperado = 1;

        const retornado = subtracao3(-2, -3);

        expect(retornado).toBe(esperado);

    })

})

* Nos três testes os valores retornados se mostraram ser o mesmo do esperado, respectivamente: -1, 5 e 1
* Tive que ter cuidado aqui porque eu quase esqueci das regras subtração quando se trata de números negativos

**Funções para verificar se um número é par**

const P\_I\_positivo = (par , impar) => {

    return [par % 2 === 0, impar % 2 === 0]

}

console.log(P\_I\_positivo(4, 3))

const P\_I\_negativo = (par , impar) => {

    return [par % 2 === 0, impar % 2 === 0]

}

console.log(P\_I\_negativo(-4, -3))

const P\_I\_zero = (zero) => {

    return zero % 2 === 0

}

console.log(P\_I\_zero(0))

describe('par ou impar', () => {

    it('Os números positivos devem retornar par(true) e impar(false)', () => {

        const esperado = [true, false]

        const retornado = P\_I\_positivo(4, 3)

        expect(retornado).toEqual(esperado);

    })

    it('Os números negativos devem retornar par(true) e impar(false)', () => {

        const esperado = [true, false]

        const retornado = P\_I\_negativo(-4, -3)

        expect(retornado).toEqual(esperado);

    })

    it('O número zero deve retornar par(true)', () => {

        const esperado = true

        const retornado = P\_I\_zero(0)

        expect(retornado).toEqual(esperado);

    })

})

* Os três testes foram um pouco diferentes porque é necessário fazer um array para retornar múltiplos valores e mudar o toBe para toEqual
* Os três testes foram um sucesso e retornaram os valores esperados:

[true, false], [true, false] e true

**Função de cálculo de fatorial**

const Fatorial = (numero) => {

    let fatorial = 1;

    for (let i = 1; i <= numero; i++) {

        fatorial \*= i;

    }

    return fatorial;

};

describe('mostre os fatoriais', () => {

    it('deve retornar o fatorial 6 de 3', () => {

        const esperado = 6;

        const retornado = Fatorial(3);

        expect(retornado).toBe(esperado);

    })

    it('deve retornar o fatorial 120 de 5', () => {

        const esperado = 120;

        const retornado = Fatorial(5);

        expect(retornado).toBe(esperado);

    })

    it('deve retornar o fatorial 1 de 0', () => {

        const esperado = 1;

        const retornado = Fatorial(0);

        expect(retornado).toBe(esperado);

    })

})

* Nos três testes os fatoriais do números apresentados foram mostrados com sucesso: 6, 120 e 1

**Função de verificação de palíndromos**

const ehPalindromo = (str) => {

    // Remover espaços, pontuações e deixar tudo em minúsculas

    const limpaString = str.toLowerCase().replace(/[^a-z0-9]/g, "");

    // Verificar se a string é igual à sua reversa

    return limpaString === limpaString.split("").reverse().join("");

};

describe('palindromos', () => {

    it('deve retornar true para ana', () => {

        const esperado = true;

        const retornado = ehPalindromo("ana");

        expect(retornado).toBe(esperado);

    })

    it('deve retornar true para radar', () => {

        const esperado = true;

        const retornado = ehPalindromo("radar");

        expect(retornado).toBe(esperado);

    })

    it('deve retornar true para "A man a plan a canal Panama"', () => {

        const esperado = true;

        const retornado = ehPalindromo("A man a plan a canal Panama");

        expect(retornado).toBe(esperado);

    })

    it('deve retornar false para casa', () => {

        const esperado = false;

        const retornado = ehPalindromo("casa");

        expect(retornado).toBe(esperado);

    })

})

* nos quatro testes realizados, os três primeiros retornaram true e o quarto retornou false, com foi esperado

**Função para encontrar o maior número em um array**

const encontrarMaiorNumero = (numeros) => {

    if (numeros.length === 0) {

        return "O array está vazio.";

    }

    return Math.max(...numeros);

};

describe ('encontrarMaiorNumero', () => {

    it('deve retornar o numero 4', () => {

        const numeros = [1, 3, 2, 4];

        const maiorNumero = encontrarMaiorNumero(numeros);

        expect(maiorNumero).toBe(4);

    });

    it('deve retornar o numero -1', () => {

        const numeros = [-1, -2, -3];

        const maiorNumero = encontrarMaiorNumero(numeros);

        expect(maiorNumero).toBe(-1);

    });

    it('deve retornar o numero 6', () => {

        const numeros = [6];

        const maiorNumero = encontrarMaiorNumero(numeros);

        expect(maiorNumero).toBe(6);

    });

});

* nos três testes, os dois primeiros retornaram 4 e -1 respectivamente e o ultimo retornou só o 6 por ser um único elemento no array

**Função de soma de todos os elementos de um array**

const somaArray = (numeros) => {

    if (numeros.length === 0) {

        return 0; // Retorna 0 se o array estiver vazio

    }

    return numeros.reduce((acumulador, valorAtual) => acumulador + valorAtual, 0);

};

describe('mostre as somas', () => {

    it('deve retornar a soma de [1, 2, 3, 4]', () => {

        const esperado = 10 ;

        const retornado = somaArray([1, 2, 3, 4]);

        expect(retornado).toBe(esperado);

    })

    it('deve retornar a soma de [-1, -2, -3, -4]', () => {

        const esperado = -10 ;

        const retornado = somaArray([-1, -2, -3, -4]);

        expect(retornado).toBe(esperado);

    })

    it('deve retornar 0', () => {

        const esperado = 0 ;

        const retornado = somaArray([]);

        expect(retornado).toBe(esperado);

    })

})

* os resultados dos testes retornaram os valores esperados: 10, -10 e 0

**Função de conversão de temperatura (Celsius para Fahrenheit)**

const celsiusParaFahrenheit = (celsius) => {

    return (celsius \* 9/5) + 32;

};

describe("celsiusParaFahrenheit", () => {

    it("deve converter 0°C para 32°F", () => {

        const celsius = 0;

        const fahrenheit = celsiusParaFahrenheit(celsius);

        expect(fahrenheit).toBe(32);

        if (fahrenheit === 32) {

            console.log(fahrenheit);

            console.log("Teste passou");

        }

    });

    it("deve converter 100°C para 212°F", () => {

        const celsius = 100;

        const fahrenheit = celsiusParaFahrenheit(celsius);

        expect(fahrenheit).toBe(212);

        if (fahrenheit === 212) {

            console.log(fahrenheit);

            console.log("Teste passou");

        }

    })

    it("deve converter -10°C para F", () => {

        const celsius = -10;

        const fahrenheit = celsiusParaFahrenheit(celsius);

        expect(fahrenheit).toBe(14);

        if (fahrenheit === 14) {

            console.log(fahrenheit);

            console.log("Teste passou");

        }

    })

})

* os testes converteram celsius para fahrenheit, com a função: (celsius \* 9/5) + 32, dando respectivamente: 32, 212 e 14

**Função que retorna os números primos até um número dado**

const numerosPrimosAte = (n) => {

    if (n < 2) return []; // Não há primos menores que 2

    const primos = [];

    for (let i = 2; i <= n; i++) {

        let ehPrimo = true;

        // Verifica se 'i' é divisível por algum número menor que ele

        for (let j = 2; j <= Math.sqrt(i); j++) {

            if (i % j === 0) {

                ehPrimo = false;

                break;

            }

        }

        if (ehPrimo) {

            primos.push(i);

        }

    }

    return primos;

};

describe('numerosPrimosAte', () => {

    it('teste com numero pequeno', () => {

        const esperado = [2, 3, 5, 7];

        const retornado = numerosPrimosAte(10);

        expect(retornado).toEqual(esperado);

        console.log(retornado);

    })

    it('teste com numero grande', () => {

        const esperado = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29];

        const retornado = numerosPrimosAte(30);

        expect(retornado).toEqual(esperado);

        console.log(retornado);

    })

    it('teste com numero grande primo', () => {

        const esperado = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97];

        const retornado = numerosPrimosAte(97);

        expect(retornado).toEqual(esperado);

        console.log(retornado);

    })

})

* os três testes retornaram os valores esperados, e da mesma forma que no teste de par ou impar, tive que usar o toEqual no lugar de toBe

**Função de manipulação de strings (inverter palavras)**

const inverterPalavras = (str) => {

    return str.split(" ").reverse().join(" ");

};

describe('inverterPalavras sem inverter as letras', () => {

    it('deve inverter as palavras', () => {

        const esperado = "mundo Olá";

        const retornado = inverterPalavras("Olá mundo");

        expect(retornado).toBe(esperado);

        console.log(retornado);

    })

    it('deve inverter as palavras', () => {

        const esperado = "incrível é JavaScript";

        const retornado = inverterPalavras("JavaScript é incrível");

        expect(retornado).toBe(esperado);

        console.log(retornado);

    })

    it('deve inverter(?) a palavra', () => {

        const esperado = "dev";

        const retornado = inverterPalavras("dev");

        expect(retornado).toBe(esperado);

        console.log(retornado);

    })

})

* a função split(" ").reverse().join(" ") inverte a ordem das palavras sem inverter a letra e retorna as palavras de ordem invertidas

**Função de calcular a média de um array de números**

const calcularMedia = (numeros) => {

    const soma = numeros.reduce((acumulador, valorAtual) => acumulador + valorAtual, 0);

    return soma / numeros.length;

};

describe('deve mostrar as médias dos array dados', () => {

    it('deve retornar a média de [1, 2, 3, 4]', () => {

        const esperado = 2.5;

        const retornado = calcularMedia([1, 2, 3, 4]);

        expect(retornado).toBe(esperado);

        console.log(retornado);

    })

    it('deve retornar a média de [-1, -2, -3]', () => {

        const esperado = -2;

        const retornado = calcularMedia([-1, -2, -3]);

        expect(retornado).toBe(esperado);

        console.log(retornado);

    })

    it('deve retornar NaN', () => {

        const esperado = NaN;

        const retornado = calcularMedia([]);

        expect(retornado).toBe(esperado);

        console.log(retornado);

    })

})

**Função que retorna o número de ocorrências de uma palavra em uma string**

const contarPalavras = (texto, palavra) => {

    // Expressão regular para encontrar todas as ocorrências da palavra

    const regex = new RegExp(`\\b${palavra}\\b`, 'gi');

    const ocorrencias = texto.match(regex);

    return ocorrencias ? ocorrencias.length : 0;

};

describe('contar quantas vezes uma palavra de uma string aparece', () => {

    it('deve contar quantas vezes a palavra javascript aparece', () => {

        const esperado = 2;

        const retornado = contarPalavras("eu gosto de javascript, javascript é legal", "javascript");

        expect(retornado).toBe(esperado);

        console.log(retornado);

    })

    it('procurando uma palavra que nao aparece', () => {

        const esperado = 0;

        const retornado = contarPalavras("javascript é legal", "node");

        expect(retornado).toBe(esperado);

        console.log(retornado);

    })

    it('Testar com palavras maiúsculas e minúsculas', () => {

        const esperado = 3;

        const retornado = contarPalavras("Aaa aaa Aaa", "aaa");

        expect(retornado).toBe(esperado);

        console.log(retornado);

    })

})

**Função para ordenar um array de números em ordem crescente**

const ordenarArrayCrescente = (numeros) => {

    return numeros.sort((a, b) => a - b);

};

describe('ordenarArrayCrescente', () => {

    it('deve ordenar um array de numeros em ordem crescente', () => {

        const numeros = [5, 3, 8, 2, 1, 4, 7, 6];

        const esperado = [1, 2, 3, 4, 5, 6, 7, 8];

        const retornado = ordenarArrayCrescente(numeros);

        expect(retornado).toEqual(esperado);

    })

    it('deve ordenar um array de numeros em ordem crescente(?)', () => {

        const numeros = [1, 2, 3, 4, 5, 6, 7, 8];

        const esperado = [1, 2, 3, 4, 5, 6, 7, 8];

        const retornado = ordenarArrayCrescente(numeros);

        expect(retornado).toEqual(esperado);

    })

    it('deve ordenar um array de numeros negativos em ordem crescente', () => {

        const numeros = [-5, -3, -8, -2, -1, -4, -7, -6];

        const esperado = [-8, -7, -6, -5, -4, -3, -2, -1];

        const retornado = ordenarArrayCrescente(numeros);

        expect(retornado).toEqual(esperado);

    })

})