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