Imutabilidade

Propriedade onde os elementos não podem ter seu valor alterado como resultado de qualquer ação por parte de expressões ou aplicação de funções. Isso protege o código de alterações indesejadas ou que passaram despercebidas e que provoquem potenciais efeitos colaterais.

Um tipo de elemento que é usualmente mutável em grande parte das linguagens de programação são os *arrays* (listas).

Solução: uso de cópia

Elementos **internos** de uma lista são mutáveis, apesar do uso do const. Evita-se o problema efetuando a operação desejada numa cópia da lista original.

```
1  // Elementos da lista são mutáveis, apesar do uso do const
2  const lista1 = [3, 1, 7, 9, 4, 6]
3  console.log(`Mutavel lista: ${lista1}`)
4
5  const lista2 = lista1.sort((a,b)=>a-b)
6  console.log(`lista (ordenada): ${lista2}`)
7
8  console.log(`lista (original): ${lista1}`)
9
10 // Corrigindo com cópia do elemento
11 const lista3 = [3, 1, 7, 9, 4, 6]
12 console.log(`Imutavel lista: ${lista3}`)
13
14 const lista4 = [...lista3].sort((a,b)=>a-b)
15 console.log(`lista ordenada: ${lista4}`)
16
17 console.log(`lista (original): ${lista3}`)
```

Proteção da lista com uso de congelamento

Uma alternativa à cópia é realizar o congelamento da lista através da função pré-definida `Object.freeze()` .

```
// Object.freeze para congelar valores de uma lista
const listacongelada = Object.freeze([3, 1, 7, 9, 4, 6])

// Tentativa de ordenar sem uso de cópia
const listaordenada = listacongelada.sort((a, b) => a - b)
console.log(listacongelada, listaordenada)

// Com uso de cópia funciona
const listaordenada = [...listacongelada].sort((a, b) => a - b)
console.log(listacongelada, listaordenada)

// Acessar partes da lista congelada funciona normalmente
const pedaco = listacongelada.slice(3)
console.log(pedaco, listacongelada)
```

⚠️ **Operações com `map`, `filter` e `reduce` não causam problemas com o princípio da Imutabilidade pois essas operações geram cópias da lista original como resultado.**

Imutabilidade para registros

Outra estrutura que sofre com esse aspecto e é preciso muita atenção em várias linguagens de programação são os *registros* (ou *objetos*).

```
const pessoa = {
    nome: 'Fulano',
    altura: 1.70,
    cidade: 'Aracaju',
    endereco: {
        rua: 'B',
        num: 306
    }
}

// Cópia por REFERÊNCIA: mutável!!!
const outro = pessoa

outro.nome = 'Beltrano'
outro.altura = 1.75

console.log(outro)
console.log(pessoa)
```

Solução: congelamento e cópia por valor

```
const pessoa = Object.freeze({
    nome: 'Fulano',
    altura: 1.70,
    cidade: 'Aracaju',
    endereco: Object.freeze({
        rua: 'B',
        num: 306
    })
})

// Alteração não efetuada: excelente!
const outro = pessoa
outro.nome = 'Beltrano'
outro.altura = 1.75
console.log('Primeira tentativa: não altera')
console.log(outro)
console.log(pessoa)

// Copia por VALOR
const maisoutro = {...pessoa}
maisoutro.nome = 'Beltrano'
maisoutro.altura = 1.75
console.log('\nSegunda tentativa: ok!')
console.log(maisoutro)
console.log(pessoa)
```

⚠️ **Atente ao fato de que o `freeze` deve ser aplicado a todas as instâncias aninhadas dos registros/objetos que eventualmente existam em sua estrutura original. Caso contrário, aquela instância "desprotegida" será mutável!**