Pureza

Uma função **pura** é uma função em que o valor de retorno é determinado APENAS por seus valores de entrada, sem efeitos colaterais observáveis.

Constantes globais vs. locais vs. parâmetros

Uso de constantes definidas globalmente no código gera funções *impuras* e *instáveis*.

```
1 //Constante global
2 const PI = 3.141592
3
4 // Impura: se alguém modificar valor de PI,
5 // mesma função gerará valor diferente
6 function areaCirc(raio) {
7   return raio * raio * PI
8 }
9 console.log(`Impura: ${areaCirc(10)}`)
```

Uso de constantes pré definidas pela própria linguagem gera funções *impuras*, porém, *estáveis*.

```
1 // Impura, mas Estável
2 // Modificar valor de Math.PI é bem mais raro
3 function areaCirc2(raio) {
4   return raio * raio * Math.PI
5 }
6 console.log(`Impura estável: ${areaCirc2(10)}`)
```

Constante definida na própria função promove *pureza*.

```
// Pura: depende apenas dos argumentos passados
// Usa parâmetros incializados
function areaCirc3(raio, pi=3.14) {
  return raio * raio * pi
}
console.log(`Pura param: ${areaCirc3(10)}`)
console.log(`Pura param: ${areaCirc3(10, 3.141592)}`)
console.log(`Pura param: ${areaCirc3(10, Math.PI)}`)

// Pura: depende apenas dos argumentos passados
// Usa constante local
function areaCirc4(raio) {
  const PI = 3.14
  return raio * raio * PI
}
console.log(`Pura local: ${areaCirc4(10)}`)
```

Valores aleatórios

Função com geração aleatória é naturalmente impura. A cada execução, o resultado será diferente.

```
function gerarNumeroAleatorio(min, max) {
    const fator = max - min + 1
    return parseInt(Math.random() * fator) + min
}

console.log(gerarNumeroAleatorio(1, 10000))
console.log(gerarNumeroAleatorio(1, 10000))
console.log(gerarNumeroAleatorio(1, 10000))
console.log(gerarNumeroAleatorio(1, 10000))
console.log(gerarNumeroAleatorio(1, 10000))
```

Efeitos colaterais observáveis

Observe o uso de `let` (ou `var`) ao invés de `const`. O conceito de variáveis para representar valores na memória computacional é típico em linguagens de programação que não seguem estritamente o Paradigma Funcional.

```
let qtde = 0

// Impura
function somar(a, b) {
    qtde++ // efeitos colaterais observáveis
    return a + b
}

// Impura
function imprimeQtde(valor) {
    console.log(`Qtde: ${valor}`)
}

imprimeQtde(qtde)
console.log(somar(68, 31))
console.log(somar(68, 31))
console.log(somar(68, 31))
console.log(somar(68, 31))
console.log(somar(68, 31))
imprimeQtde(qtde)
```