```
//FUNÇÕES ANÔNIMAS:
//Funções anônimas são funções que não recebem um nome explicitamente. Como por exemplo as arrow function que são sempre anônimas.
//Geralmente atribuímos funções anônimas a variáveis e constantes, ou as aplicamos diretamente dentro de outras funções.

//EXEMPLO DE FUNÇÃO ANÔNIMA:
const somaDoisNumeros = function (a, b) {//Perceba que a própria função não possuí um nome, por isso, fomos obrigados á colocá-
la em uma variável somente para poder referenciá-la futuramente...
    return a + b;
}
console.log(`1) Perceba que a função funciona perfeitamente mesmo sem ter um nome:`)
console.log(somaDoisNumeros(2, 3))



//PASSANDO UMA FUNÇÃO ANÔNIMA DENTRO DE OUTRA FUNÇÃO ANÔNIMA:
const colocaDoisNumerosEEscolheOperacao = (a, b, operacao = somaDoisNumeros) => { //Perceba que temos uma arrow function, que é uma f
unção anônima por padrão, e dentro dela temos um parâmetro que recebe uma função, e se não receber nenhuma, usa a função "somaDoisNum
eros" por padrão...
    return operacao(a, b);
}
console.log(`\n2) Função que recebe outra função dentro dela:`)
console.log(colocaDoisNumerosEEscolheOperacao(4, 4));



//PASSANDO UMA FUNÇÃO ANÔNIMA DURANTE A CHAMADA DE UMA FUNÇÃO:
console.log(`\n3) Passamos uma função anônima durante a chamada de uma função:`)
console.log(colocaDoisNumerosEEscolheOperacao(50, 20, (x, y) => x - y)); //Perceba que passamos uma arrow function dentro de uma funç
ão durante a chamada dessa função, com uma operação de subtração...
```

RESULTADO NO CONSOLE...

```
[Running] node "c:\Users\User\Documents\JAVASCRIPT\ARQUIVOS_DAS_AULAS\068-Funcoes_anonimas.js"
1) Perceba que a função funciona perfeitamente mesmo sem ter um nome:
5
```

```
2) Função que recebe outra função dentro dela:
8

3) Passamos uma função anônima durante a chamada de uma função:
30

[Done] exited with code=0 in 0.136 seconds
```