Definição de função como retorno

Se uma função é um valor/expressão então ela deve herdar todas as características deste, incluindo **poder ser retornada como resultado** da aplicação de outra função.

Vamos generalizar o problema anterior de calcular a quarta potência de um número para calcular uma potência qualquer, fornecida pelo interessado (ex. $x^2, x^4, x^{\frac{1}{2}}, x^{-2}$), ou seja, x^n. Observe que temos dois argumentos (x e n) a serem passados a uma eventual função que resolva o problema.

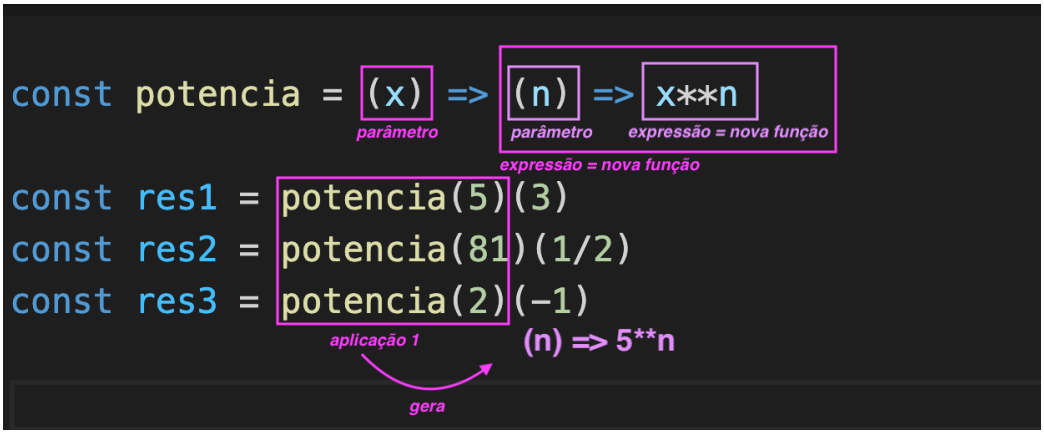
Casos como esse podem ser resolvidos usando a definição clássica de função: `fpotencia = (x,n) => x**n`

Entretanto, uma notação mais flexível e consoante com a definição de função enquanto valor/expressão é **desagregar os parâmetros** e, assim, viabilizar **aplicação independente dos argumentos**.

[EXEMPLO] Programa para calcular a potência de um número.

OBS: a partir de agora e para o resto do curso, iremos excluir a letra `f` inicial da nomenclatura das funções, uma vez que, conforme explicado, funções devem ser tratadas indistintivamente do conceito de valores/expressões.

```
1  const potencia = (x) => (n) => x**n
2
3  const res1 = potencia(5)(3)
4  const res2 = potencia(81)(1/2)
5  const res3 = potencia(2)(-1)
6
7  console.log(res1)
8  console.log(res2)
9  console.log(res3)
```



[EXEMPLO] Defina funções para calcular o `quadrado`, o `cubo` e a `raiz quadrada` de um número passado como argumento reaproveitando uma definição de função genérica chamada `expoente`.

"LEIA DE BAIXO PARA CIMA"

```
1 const expoente = (e) => (base) => base**e
2
3 const quadrado = expoente(2)
4 const cubo = expoente(3)
5 const raizq = expoente(1/2)
6
7 console.log(quadrado(10))
8 console.log(cubo(3))
9 console.log(raizq(81))
```