

# Funções

**DigitalHouse** >  
Coding School



**Certified Tech  
Developer**  
The Ultimate Degree

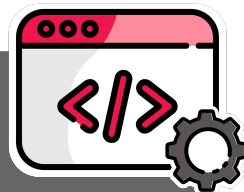
# Índice

1. [Retomando funções](#)
2. [Arrow functions](#)
3. [Funções como parâmetros](#)

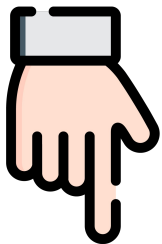
# 1 | Retomando funções

# O que é uma **função**?

```
function somar(par1, par2) {  
  
    let valor = par1, par2;  
  
    return valor;  
  
}  
  
somar(1, 3);
```



O JavaScript tem estruturas chamadas funções, que são apenas trechos de código, os quais **não** se executam até que, em algum momento, o “invocamos” ou lhe chamamos com um nome ou identificador.



```
function somar(par1, par2) {  
    let valor = par1, par2;  
  
    return valor;  
  
}  
  
somar(1, 3);
```

Aqui utilizamos a palavra-chave **function** para indicar uma declaração de uma função chamada **soma** que recebe dois parâmetros **par1** e **par2**.



```
function somar(par1, par2) {
```

```
    let valor = par1, par2;
```

```
    return valor;
```

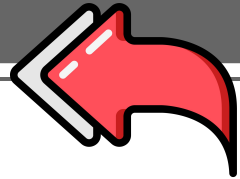
```
}
```

```
somar(1, 3);
```

Definimos o que a função faz,  
pode ser qualquer algoritmo que  
nos seja útil.

```
function somar(par1, par2) {  
  
    let valor = par1, par2;  
  
    return valor;  
  
}  
  
somar(1, 3);
```

Com o **return**, saímos da função e voltamos para a linha onde foi chamada. Também podemos devolver um **valor**, ou simplesmente não devolver, por meio do uso do **return**; apenas.



```
function somar(par1, par2) {  
  
    let valor = par1, par2;  
  
    return valor;  
  
}
```

```
somar(1, 3);
```



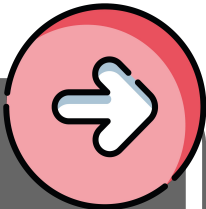
Chamamos ou invocamos a função **somar** com os parâmetros **1** e **3**.



# 2 | Arrow functions

# Arrow functions

```
let somar = (a, b) => {  
    let valor = a + b;  
    return valor;  
}  
  
somar(1, 3);
```



Existe outra forma de declarar uma função utilizando a notação de '**arrow function** (**=>**)'. Esta função se comporta da mesma maneira que as funções normais de Javascript e também se invocam da mesma forma.

# Algumas particularidades

No seguinte trecho de código, podemos ver uma arrow function que faz exatamente o mesmo que nos exemplos anteriores.

## Nome de função

Neste caso, a função tem o nome da variável a qual está atribuída.

## Return rápido

Esta forma de declarar uma função torna possível o retorno sem utilizar a palavra return, e isto acontece quando não utilizamos chaves para encerrar o nosso algoritmo. Isto geralmente só é utilizado quando pode ser expresso numa única linha.

JS

```
let somar = (a, b) => a + b;
```

Parâmetros

# 3 | Funções como parâmetros

# Funções como parâmetros

O JavaScript nos permite utilizar uma função **como parâmetro de outra**. Isto é muito útil para o nosso código:

```
{  
  function executor(func) {  
    // código da função  
    func(1, 2);  
    // código da função  
  }  
  function somar(a, b) {  
    return a + b;  
  }  
  executor(somar);  
}
```





No exemplo, pode se ver que temos uma função **executor**, que executa todo o seu algoritmo e, quando é necessário, executa a função **func** passado como um parâmetro, que corresponde a função **soma**. Nesses casos, diz-se frequentemente que a função de **executor** é a responsável de executar a função **somar** (ou qualquer outra que lhe passamos).

DigitalHouse>  
Coding School