

# Programação de IAs Conversacionais

## *Módulo 1 - Funções*

---



## Índice

1. Introdução
2. Declaração de funções
3. Criar funções
4. Funções parametrizadas
5. Arrow Functions





*Programação de IAs Conversacionais*

# Introdução



# O que é uma função

Funções soluções reutilizáveis

No nosso dia a dia aplicamos o conceito de funções sem perceber

Ações repetitivas são como as funções

Dividimos um problema grande em pequenas partes para facilitar a solução

# Funções em programação

Em programação usamos as funções quando precisamos repetir uma ação várias vezes

Por exemplo a calculadora usa funções para calcular

Função soma:

```
function soma(){  
    console.log(2 + 5)  
}
```

```
// irá imprimir 7
```



*Programação de IAs Conversacionais*

# Declaração de funções



# Sintaxe

Dentro das funções colocamos as instruções que desejamos

```
function bomDia(){  
    console.log("Olá! Tenha um bom dia!")  
}
```

Primeiramente devemos usar a palavra reservada **function**

Depois escolhemos um nome para nossa função para “chamarmos” ela depois

Sempre é preciso colocar o parênteses

# Sintaxe

```
function bomDia(){  
    console.log("Olá! Tenha um bom dia!")  
}
```

Depois dos parênteses vem as chaves { } as instruções que desejamos fazer ficam dentro deste bloco

Podemos criar variáveis, loops, qualquer coisa, mas sempre com pensamento de resolver um problema





*Programação de IAs Conversacionais*

# Criar Funções



# Exemplo de uma função

Essa função usa variáveis e loop para imprimir na tela a nossa lista de compras:

```
function imprimirLista(){  
  
    var listaDeCompras = ['Batata', 'Maçã', 'Leite',  
    'Chocolate']  
  
    for(var i = 0; i < listaDeCompras.length; i++){  
        console.log("O item da lista é: " +  
        listaDeCompras[i])  
    }  
}
```



*Programação de IAs Conversacionais*

# Funções parametrizadas



# Funções com parâmetros

Vimos a função somar:

```
function soma(){  
    console.log(2 + 5)  
}
```

Essa é uma função reutilizável?

E se quisermos somar outros números?

# Funções com parâmetros

Podemos passar dados como parâmetros!

Os **parâmetros** são informações que passamos dentro dos parênteses das funções

São valores dinâmicos

Se comportam como variáveis dentro da nossa função

# Chamando as funções

Exemplo:

```
function soma(num_1, num_2){  
    console.log(num_1 + num_2)  
}
```

Para executar essa função devemos “chama-la”:

```
soma(5,7) //irá imprimir 12
```

# Retornando o resultado

Vimos que a nossa função imprime o resultado na tela

Mas e se quisermos usar esse resultado para outra coisa?

Precisamos usar uma outra palavra reservada: **return**

Como o nome já diz, irá **retornar** o resultado da nossa função, pode ser qualquer valor ou tipo de dado

```
function soma(num_1, num_2){  
    return num_1 + num_2  
}
```

# Retornando o resultado

O uso do return não é obrigatório

A execução da função para quando chega no return:

```
function soma(num_1, num_2){  
    return num_1 + num_2  
    console.log(num_1 + num_2)  
}
```

//não irá imprimir na tela pois o console.log está depois do return



# Retornando o resultado

É importante entender o que a sua função vai fazer

Imprimir na tela ou retornar um resultado para ser usado depois?



*Programação de IAs Conversacionais*

# Usando as funções



# Funções sem params

```
function soma(){  
    console.log(2 + 5)  
}
```

Para chamar uma função sem parâmetro usamos o nome e os parênteses:

```
soma()
```

Não usamos a palavra reservada function novamente pois somente a usamos para criar as funções

# Funções com params

```
function soma(num_1, num_2){  
    return num_1 + num_2  
}
```

Para chamar uma função com parâmetro usamos o nome e dentro dos parâmetros os valores

`soma(12, 8) //retornará 20`

Veja que os dados que passamos no momento que chamamos a função devem seguir a ordem do parâmetro. num\_1 virou 12 e num\_2 virou 20

# Sequência dos params

Veja a importância da sequência de atribuição dos dados na sequência dos params

```
function divisao(num_1, num_2){  
    return num_1 / num_2  
}
```

`divisao(12, 3)` //retornará 4

Se trocarmos a sequência:

`divisao(3,12)` //retornará 0.25

# Como guardar o resultado da função

Usamos as **variáveis**

```
function soma(num_1, num_2){  
    return num_1 + num_2  
}
```

```
var resultado = soma(2,7)
```

```
console.log(resultado) //imprime 9
```



*Programação de IAs Conversacionais*

# Arrow Functions



# Arrow o que?

Usamos muito as funções na programação

As **arrow functions** são funções também mas com uma sintaxe mais compacta

```
function soma(a, b){  
    return a + b  
}
```

Numa arrow function ficará assim:

```
var somar = (a, b) => a + b
```



# Sintaxe

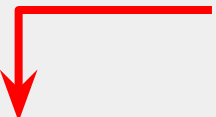
Mas e o nome da função?

As arrow functions não tem nome, são **anônimas**

Por isso sempre declaramos uma variável para guardar o resultado da função

```
var resultado = (a, b) => a * b;
```


# Sintaxe

 Variável para guardar o resultado

```
var resultado = (a, b) => a * b;
```


# Sintaxe

Atribuição




```
var resultado = (a, b) => a * b;
```

# Sintaxe

var resultado =  Parâmetros  
`(a, b) => a * b;`

# Sintaxe

var resultado = (a, b)  => a \* b;

Operador flecha, substitui a palavra  
function

# Sintaxe

```
var resultado = (a, b) => a * b;
```

Se a função tem apenas uma linha não precisamos usar as chaves

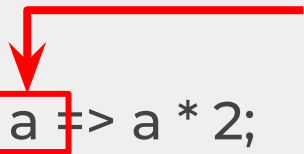
```
var resultado = (a, b) => {  
    return a * b  
}
```

Caso contrário precisa das chaves e do return

# Sintaxe

```
var resultado = a => a * 2;
```

Se tiver apenas um parâmetro não precisamos dos parênteses



# Exemplo

```
var somaDois = valor1 => valor1 + 2;
```

```
var multiplicacao = (a, b) => a * b;
```

```
var concatenaNomes = ( ) => {  
    var nome = "Alice"  
    var sobrenome = "Silva"  
    return nome + sobrenome  
}
```





*Programação de IAs Conversacionais*

# Exercícios



# Exercícios

Crie uma função que receba um valor e informe se é par ou ímpar

```
function ehParOuImpar(num1){  
  if (num1 % 2 == 0){  
    return "É par"  
  } else {  
    return "É ímpar"  
  }  
}
```

```
var num_1 = 3
```

```
ehParOuImpar(num_1)
```

# Exercícios

Crie uma função que receba dois números e informe se a soma destes números é par ou ímpar