

JavaScript: Funções



DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

- Funções
 - O que são?
 - Como declaramos?
 - Parâmetros
 - Retorno
 - Boas práticas

FUNÇÕES - O QUE SÃO?



FUNÇÕES - O QUE SÃO?

Repetição de código?

Alguma vez você já precisou **repetir muito** um código?

#exemplo: Um banco precisa converter o valor recebido em dólar para real

```
// Banco - converter dólar para real
// cotação do dólar hoje = 5.2

const valorEmDolar = 100
const valorEmReal = valorEmDolar * 5.2

const valorEmDolar2 = 200
const valorEmReal2 = valorEmDolar2 * 5.2
```

```
function calcularValorEmReais(cotacao, valorEmDolar) {
  console.log(cotacao * valorEmDolar)
}
```

```
calcularValorEmReais(5.2, 100)
calcularValorEmReais(5.2, 200)
```

Solução = Função

A função é uma sintaxe do JS que permite **englobar** um conjunto de instruções em um só lugar, substituindo os procedimentos repetitivos.

FUNÇÕES - COMO DECLARAMOS?

Funções nomeadas

Funções declaradas com o uso da palavra reservada **function**.

```
function nomeDaFuncao(parametros){  
  //instruções a serem executas  
}  
  
// === Exemplo ===  
// Declaração  
function cumprimentar(){  
  console.log('oi')  
}  
// Invocação  
cumprimentar()
```

```
const variavel = function (parametros){  
  //instruções a serem executas  
}  
  
// === Exemplo ===  
// Declaração  
const cumprimentar = function(){  
  console.log('oi')  
}  
// Invocação  
cumprimentar()
```

Funções não-nomeadas

São funções associadas diretamente a uma **variável**.

FUNÇÕES - COMO DECLARAMOS?

Na prática, tem diferença?

Sim!!!!

Função não-nomeada só pode ser **invocada** depois da declaração.

```
29 cumprimentar()
30 function cumprimentar(){
31   console.log('oi')
32 }
33
```

PROBLEMAS SAÍDA TERMINAL GITLENS

oi

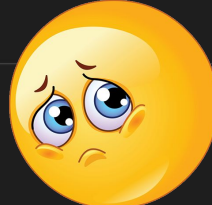


```
36 cumprimentar()
37 const cumprimentar = function(){
38   console.log('oi')
39 }
40
```

PROBLEMAS SAÍDA TERMINAL GITLENS COMMENTS

cumprimentar()
^

ReferenceError: Cannot access 'cumprimentar' before initialization



Transforme o código abaixo em uma função

```
const somaDeDoisNumeros = 1 + 2
console.log(somarDoisNumeros) // imprime 3 na tela

const somaDeDoisNumerosB = 3 + 4
console.log(somaDeDoisNumerosB) // imprime 7 na tela
```

FUNÇÕES - PARÂMETROS

Parâmetros

Para que a função possa ser reutilizável
(como vimos lá no começo da aula),
ela pode receber parâmetros
(equivalentes a variáveis)
que serão adaptados caso a caso.

```
function nomeDaFuncao(parametro1, parametro2){  
  //instruções a serem executas  
}  
  
// === Exemplo ===  
// Declaração  
function somarDoisNumeros(numero1, numero2){  
  console.log(numero1 + numero2)  
}  
// Invocação  
somarDoisNumeros(2, 3)
```

```
// Parâmetro padrão = undefined  
function somarDoisNumeros(numero1, numero2){  
  console.log(numero1 + numero2)  
}  
somarDoisNumeros(2) // retorna NaN  
  
// Usando um parâmetro opcional  
function somarDoisNumeros(numero1, numero2 = 0){  
  console.log(numero1 + numero2)  
}  
somarDoisNumeros(2) // retorna 2
```

Parâmetro Padrão

Em JS, por padrão os parâmetros são
undefined.
Mas em algumas situações pode ser útil
usar um **valor diferente**.

FUNÇÕES - RETORNO

Sem retorno

Como vimos até o momento, as funções podem não retornar nenhum valor, mas imprimir/**exibir o resultado** das instruções (com `console.log`) **no console**.

```
function cumprimentar(){  
  console.log('oi')  
}  
  
cumprimentar() // imprime "oi" na tela
```

```
function cumprimentar(){  
  return 'oi'  
}  
  
cumprimentar() // não imprime nada  
console.log(cumprimentar()) // imprime "oi" na tela
```

Com retorno

A palavra reservada **return** permite que a função **devolva algum valor** para onde a chamada for feita e **finalize a função** (nada será considerado após o return)

FUNÇÕES - RETORNO

+ funções com retorno

Elas **finalizam a função**, ou sejam, depois delas nada mais será considerado

```
function cumprimentar(){  
  return 'oi'  
  console.log('olá!!!') //código inalcançável!!  
}  
console.log(cumprimentar()) // imprime "oi" na tela
```

Podemos guardar o retorno em uma **variável** e usá-lo em outras partes do código.

```
function cumprimentar(){  
  return 'Oi'  
}  
const cumprimento = cumprimentar()  
const cumprimentarCarol = cumprimento + ' Carol!!!'  
console.log(cumprimentarCarol) // imprime "Oi Carol!!!" na tela
```

INTERVALO DE AULA

DEV!

Finalizamos o nosso primeiro período de hoje. Que tal descansar um pouco?!

Nos vemos em 20 minutos.

Início: 20:20

Retorno: 20:40



- O nome da função deve refletir seu propósito
- Usar verbos para nomear
- Fazer apenas uma coisa
- Seguir **camelCase**

EXERCÍCIO 1

1 - Crie uma **função não nomeada** que recebe um número e devolva uma string dizendo se é par ou ímpar.

EXERCÍCIO 2

2 - Crie uma **função nomeada** que recebe um array de elementos, imprima cada um dos elementos e em seguida retorne a quantidade de elementos no array (seu tamanho).

EXERCÍCIO 3

3 - Crie uma **função** que recebe um array de números e retorna a quantidade de números pares e a quantidade de números ímpares.

Em seguida imprima na tela uma string informando a quantidade de valores informados, a quantidade de ímpares e de pares.

(ex: A quantidade informada foi ____, a de valores pares foi __ e a de valores ímpares foi __)

MATERIAL COMPLEMENTAR

- [Funções - JavaScript | MDN](#)
- Código da aula: <https://github.com/rosanarezende/2022-turma-clamed>



DEVinHouse

Parcerias para desenvolver a sua carreira

OBRIGADO!



<LAB365>