

# REVISÃO DE FUNÇÕES, CALLBACK



DEVinHouse

Parcerias para desenvolver a sua carreira

**SENAI**

<LAB365>

# AGENDA

- Revisão de funções
- Callbacks

# REVISÃO DE FUNÇÕES

# REVISÃO DE 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.

# REVISÃO DE 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**.

# REVISÃO DE 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

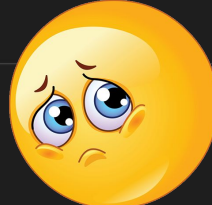


```
36 cumprimentar()
37 const cumprimentar = function(){
38   console.log('oi')
39 }
40
```

PROBLEMAS SAÍDA TERMINAL GITLENS COMMENTS

cumprimentar()  
^

ReferenceError: Cannot access 'cumprimentar' before initialization



# REVISÃO DE 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**.

# REVISÃO DE FUNÇÕES: RETORNO

## Sem retorno

Como vimos até o momento, as funções podem não **retornar nenhum valor**:

**undefined**

Por exemplo, imprimir/exibir o resultado das instruções no console.

```
function cumprimentar(){  
  console.log('oi')  
}  
  
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).

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

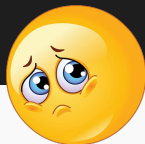


# REVISÃO DE FUNÇÕES: REAPROVEITAMENTO

Apenas nas **funções com retorno** podemos **guardar o resultado** em uma variável e **reaproveitá-lo** em outras partes do código.

```
// Quando a função não tem retorno, o resultado é undefined
const imprimeNomeDaPessoa = function(nome){
  console.log(nome)
}
const carol = imprimeNomeDaPessoa("Carol")

const cumprimentaPessoa = function(nome){
  console.log(`Olá, ${nome}`)
}
cumprimentaPessoa(carol) // Olá undefined
```



```
// Quando a função tem retorno, temos acesso ao resultado
const retornaNomeDaPessoa = function(nome){
  return nome
}
const rosana = retornaNomeDaPessoa("Rosana")

const cumprimentaPessoa = function(nome){
  console.log(`Olá, ${nome}`)
}
cumprimentaPessoa(rosana) // Olá, Rosana
```



# REVISÃO DE FUNÇÕES: Exercício 1

Crie uma função que recebe um array de strings e retorne a maior string do array.

#exemplo de array de strings:

```
const estados = ['Sao Paulo', 'Rio de Janeiro', 'Amazonas', 'Pernambuco', 'Santa  
Catarina', 'Rio Grande do Norte', 'Sergipe']
```

## REVISÃO DE FUNÇÕES: Exercício 2

Crie uma função que recebe um array e devolve seus valores de forma invertida (de trás pra frente)

# 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



# **CALLBACK**

# **CALLBACK: Porque estudar?**

**FUNÇÕES** PODEM RECEBER **VARIÁVEIS**  
*(como entradas/parâmetros)*

**VARIÁVEIS** PODEM RECEBER **FUNÇÕES**  
*(como funções não nomeadas)*

**FUNÇÕES** PODEM RECEBER **FUNÇÕES**  
*(como entradas/parâmetros)*



# CALLBACK: O que é?

Callbacks são **funções** que são passadas como **parâmetro** para outras funções.

- no exemplo a função callback será *imprimeResultado()*

Quando o evento acontece, a função é executada e **chamada de volta** (por isso callback).

```
const informaParOuImpar = function(numero, funcaoASerUtilizada) {
  let resultado
  if (numero % 2 === 0) {
    resultado = 'par'
  }
  else if (numero % 2 === 1) {
    resultado = 'impar'
  }
  else {
    resultado = 'não foi informado um valor válido'
  }
  funcaoASerUtilizada(resultado)
}

const imprimeResultado = function(resultadoEsperado) {
  console.log(`O resultado é: ${resultadoEsperado}`)
}

const comemoraSeForPar = function(resultadoEsperado) {
  if (resultadoEsperado === 'par') {
    console.log('Parabéns! Você é um bom número!')
  }
}

informaParOuImpar(2, imprimeResultado) // O resultado é: par
informaParOuImpar(2, comemoraSeForPar) // Parabéns! Você é um bom número!
```

# CALLBACK: Uso mais comum

A maior utilização de callbacks é com **funções javascript já existentes**.  
Criar funções que usem essa lógica ajudará a fortalecer o conceito  
e utilizar essas funções com êxito.

Exemplos que ***necessitam de callback***:

addEventListener(), setTimeout(), setInterval(), reduce(), map()

#exemplo com map()

```
// Forma já estudada
const resultadoDoMap = [10,20,30].map(function(numero) {
  return numero + 100
})
console.log(resultadoDoMap) // [110, 120, 130]

// Forma com callback mais explícita
const somaCom100 = function(numero) {
  return numero + 100
}
const resultadoDoMap2 = [10,20,30].map(somaCom100)
console.log(resultadoDoMap2) // [110, 120, 130]
```

#exemplo com addEventListener()

```
// Forma já estudada
botao3.addEventListener("click", function() {
  conteudo.innerHTML = "Conteudo aqui"
})

// Forma com callback mais explícita
const funcaoDeInserirConteudo = function() {
  conteudo.innerHTML = "Conteudo aqui"
}
botao3.addEventListener("click", funcaoDeInserirConteudo)
```



## **CALLBACK: Exercício**

Crie um site com um botão que espera receber um evento de click e uma função callback que informe ao usuário que o botão foi clicado.

## CALLBACK: Exercício

Crie 2 funções, ambas recebem um número e retorna true ou false:

- a primeira verifica se é par
- a segunda verifica se é ímpar

Em seguida crie uma função que recebe como parâmetros: um array de números e uma função callback.

A função principal deverá ser capaz de filtrar os números e retornar um array de números filtrados (independente de qual seja a função callback).

Dica: o filtro poderá ser de números ímpares ou pares.

## AVALIAÇÃO DOCENTE

O que você está achando das minhas aulas neste conteúdo?

[Clique aqui](#) ou escaneie o QRCode ao lado para avaliar minha aula.

Sinta-se à vontade para fornecer uma avaliação sempre que achar necessário.





# DEVinHouse

Parcerias para desenvolver a sua carreira

**OBRIGADO!**



<LAB365>