Semana 4 - Aula 13

Funções

Labenu_

Sumário

Labenu_

O que vamos ver hoje?

- Hoje, vamos continuar a falar sobre o Javascript, iniciando o conceito sobre **funções**
    - Funções nomeadas
    - Funções não-nomeadas
    - Arrow functions
- Boas práticas

O que são funções?

Labenu_

O que são funções? 🤪

- Já esteve em alguma situação em que você teve que **repetir muito um código**?

O que são funções? 🤪

- Por exemplo: um banco que permite realizar **transações internacionais** sempre precisa realizar uma **conversão entre as moedas**.

```
const valorEmDolares = 100
const valorEmReais = valorEmDolares*5.49
```

O que são funções? 🤪

- **Funções** são uma sintaxe que permite **englobar uma parte** do nosso código em um só lugar

- E, então, podemos chamá-la sempre que precisarmos

O que são funções? 🤪

- Uma função pode **receber** (ou não) argumentos/ parâmetros (**input**) e **devolver** (ou não) um resultado (**output**/result)

O que são funções? 🤪

- Resumindo...

Entrada (Input) ➡  ➡ Saída (Output)

Funções nomeadas

Labenu_

Funções nomeadas 名

- Uma **função nomeada** é uma função que é declarada com a **seguinte sintaxe**

```
function nomeDaFuncao(parâmetros){
    // tarefa a ser executada
}
```

Funções nomeadas 名

- **Exemplo**

  Declaração

```
function dizOi(){
   console.log("Oi!");
}
```

  Chamada/Invocação

```
   dizOi()
```

Vamos ver na prática! 

Funções nomeadas 名

- Como comentado, as **funções** podem receber parâmetros/argumentos/**input**

- A **sintaxe** para isto é:

```
function minhaFuncao(argumento1, argumento2){
    // tarefa a ser executada
}
```

Funções nomeadas 名

- **Exemplo**

  Declaração

```
function somaDoisNumeros(a, b){
  const soma = a+b
  console.log(soma)
}
```

  Chamada/Invocação

```
somaDoisNumeros(2, 4)
```

Funções nomeadas 名

- Existe uma **keyword** reservada para funções, que permite que ela **devolva** algum valor ou variável (**output**)

- Essa palavra é o `return`.

```
function minhaFuncao(argumento1, argumento2){
    return valor
}
```

Funções nomeadas 名

- **Exemplo**

Declaração

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

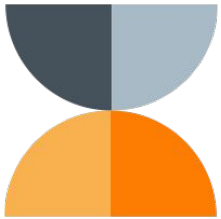
Chamada/Invocação

```
const resultado = somaDoisNumeros(2, 4)
// variável resultado possui o valor 6
```

Vamos ver na prática! 

Exercício 1 - *Funções Nomeadas*

Crie uma função que receba um array e imprima cada um dos seus elementos e devolva a quantidade de elementos nele.



Pausa para relaxar 😴

5 min

- **PONTOS IMPORTANTES:**
  - Funções são uma estrutura que permite isolar uma parte do nosso código
  - Elas podem receber parâmetros/argumentos e retornar variáveis ou valores
  - Usamos o **return** para devolver um valor ou variável

Funções não-nomeadas

Labenu_

Funções não-nomeadas 😶

- Uma função não nomeada é uma função que pode ser **associada diretamente a uma variável**

```
let variável = function(parâmetros){
  // tarefa a ser executada
}
```

Funções não-nomeadas 😶

- **Exemplo**

  Declaração

```
let somaDoisNumeros = function (a, b){
  return a+b
}
```

  Chamada/Invocação

```
const resultado = somaDoisNumeros(3, 9)
// variável resultado possui o valor 12
```

Vamos ver na prática! 🔬

Funções não-nomeadas 😶

- Existe outra sintaxe que facilita a escrita destas funções: a **arrow function**

```
let nome = (parametros) => {
  // tarefa a ser executada
}
```

Funções não-nomeadas 😶

- **Exemplo**

  Declaração

```
let somaDoisNumeros = (a, b) => {
  return a+b
}
```

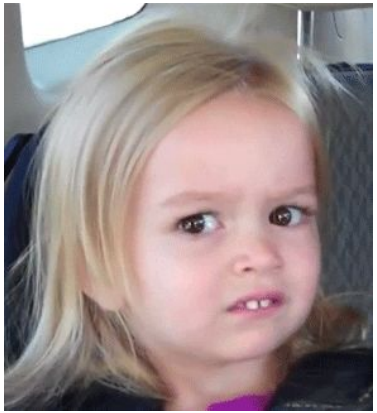
  Chamada/Invocação

```
const resultado = somaDoisNumeros(3, 9)
// variável resultado possui o valor 12
```

Vamos ver na prática! 

Funções não-nomeadas 🫥

- Se a chamada das funções é a mesma e elas funcionam da mesma forma, **qual a diferença entre elas**?



Funções não-nomeadas 😶

- **Função Nomeada VS Não Nomeada**
  - A função não nomeada **SÓ** pode ser invocada depois da sua declaração
  - A função nomeada pode ser chamada de qualquer parte do código, mesmo antes da sua declaração efetiva

Vamos ver na prática! 🔬

Exercício 2 - *Funções Não-Nomeadas*

Crie uma função que receba um array e imprima cada um dos seus elementos e devolva a quantidade de elementos nele.

Boas práticas de funções

Labenu_

Boas práticas

- Assim como variáveis, uma função deve ter um nome que reflita seu propósito
  - Verbos no infinitivo
  - camelCase https://pt.wikipedia.org/wiki/CamelCase

- A função deve fazer apenas uma coisa
  - Se for possível extrair outra função de nome significativo, está fazendo mais de uma coisa

- As funções devem estar em uma ordem lógica, na medida do possível

Resumo

Labenu_

Resumo 

- **Funções** são estruturas que permitem isolar uma parte do nosso código e reaproveitá-lo depois

- Existem as **funções nomeadas**

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

Resumo 

- Existem as **funções não nomeadas**

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

- Existem as **arrow functions**

```
let soma = (a, b) => {
  return a+b
}
```

Dúvidas?

Labenu_

Obrigado!