Aula III - Funções

Amanda Luna, Anderson Nóbrega e João Soares

Incodde, Imersão Incodde

Objetivo

Olá, pessoal. Em nossa segunda aula procuramos compreender o conceitos de Atribuições Condicionais, laços de Repetição e Github. Na aula de hoje, continuaremos a apronfundar nossos conhecimentos e estudar sobre **Funções**. Convém salientar, que é de extrema relevância termos os conceitos básicos muito bem compreendidos para que possamos nos tornar bons profissionais no mercado de trabalho.

Pois bem, na aula de hoje procuraremos compreender e trabalhar sobre **Funções**. Vamos lá? Desejamos uma excelente aula para vocês e, lembrem-se, estamos à disposição para contribuir com o vosso conhecimento.

Introdução à funções

Funções são blocos de instrução que podem ser invocados de qualquer parte do nosso código. Toda função, por definição formal, possui um nome, pode receber (parâmetros) - ou não - e pode retornar valores. É de extrema relevância lembrarmos que, nem toda função receberá parâmetros, da mesma forma, que nem toda função retornará valores. Tudo dependerá da situação, ainda assim, a estrutura sempre seguirá um padrão e por isso, é facilmente reconhecido quando nos depararmos.

As funções podem definir em seu cabeçalho um conjunto de valores que devem ser enviados quando a função for invocada. Desta forma, a função só será invocada se os valores forem definidos. Após a execução das instruções definidas no bloco de instrução da função, a função poderá retornar um valor e, dessa forma, temos um ciclo em que os dados são enviados, processados e retornados. Por isso é que raramente devemos utilizar variáveis globais, até porque, todos valores podem ser enviados e retornados pelas funções de maneira explícita.

A seguir mostraremos exemplos,

Nesse primeiro exemplo temos uma função sem retorno e com parâmetros

```
// Função sem return e com parâmetros

function adicionaFilme(nomeDoFilme, lista){
    lista.push(nomeDoFilme);
}
```

Nesse segundo exemplo temos uma função com retorno e com parâmetros

```
// Função com return e com parâmetros
function soma(a, b) {
        return a + b;
}
```

Arrow Functions

Apesar de ser uma feature bastante usada hoje em dia, algumas pessoas ainda não conhecem alguns segredos dessa nova sintaxe de funções em JavaScript.

- Nova sintaxe

Pra quem ainda não teve contato com as arrow functions, Vamos começar entendendo a nova sintaxe.

```
const sum = (a, b) => {
  return a + b
}
```

A primeira vista pode parecer estranho, mas usando **function** tradicional, podemos obter o mesmo resultado assim:

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

A criação de uma *arrow function* consistem em 3 "passos":

- Os parênteses, que é por onde a função recebe os argumentos (assim como na **function** tradicional);
- A "seta" => - responsável pelo nome "arrow" function;
- E as chaves: o bloco de código que representa o corpo da função.

Perceba também que nós usamos a arrow function como uma **expressão** - no caso, atribuindo o resultado da criação da arrow function à uma variável.

Diferente das **function** tradicionais, não existe uma forma literal de escrever uma arrow- function. Ela sempre deve ser usada como uma **expressão**. Também não é possível nomear uma arrow function. Elas são sempre anônimas.

Desafio 03

Descrição:Dado 2 valores inteiros X e Y. A seguir, calcule e mostre a soma dos números impares entre eles. O programa deve imprimir um valor inteiro. Este valor é a soma dos valores ímpares que estão entre os valores fornecidos na entrada que deverá caber em um inteiro.

- 1ª entrada: X = 6 e Y = -5 saída: 5
- 2ª entrada: X = 15 e Y = 12 saída: 13
- 3ª entrada: X = 12 e Y = 12 saída: 0