Certified Tech Developer
The Ultimate Degree

Funções e procedimentos - Exercício

Sabemos que uma função busca concentrar em um único espaço uma série de etapas a serem executadas sob um nome que descreva a finalidade dessas etapas. Coloque (Vermelho), Mova (Norte), adicione (2, 2), console.log ("Olá Mundo").

Praticaremos a criação e utilização desta ferramenta na linguagem Javascript, no ambiente Node JS, no editor de texto VS Code.

Objetivo

Dominar as habilidades de desenvolvimento, ordenando nosso código em pastas e arquivos, definindo e invocando funções em JS, dando-lhes nomes descritivos, combinando-os entre si para uma finalidade mais complexa.

Exemplo: Math

Como vimos, JS já incluiu em seu código uma série de ferramentas, incluindo algumas sobre matemática. Se escrevermos para um .js o procedimento `Math.random()` (link) e a executamos, parece que nada acontece. Porém, o problema é que não pedimos a essa função que nos mostre o que ela faz pelo console: `console.log( Math.random() );`. Note como o log se encarrega de mostrar no console o que há entre seus parênteses, que neste caso é o que Math.random () faz, e que não havíamos visto antes.

Algo saiu agora no console?

Vamos executar várias vezes aquele file .js. Ele sempre dará um resultado diferente, uma vez que aleatório significa aleatório, e o que ele faz é entregar um número aleatório entre 0 e 1.

Agora, essas 2 funções, console.log () e Math.random (), não criamos, mas elas já vêm com a linguagem, basta invocá-las ou chamá-las. Eles são informados assim quando queremos gerenciá-los. Temos que criar nossas próprias funções.

Definindo e invocando uma função

Definir e criar uma função ou a mesma, neste caso vamos definir uma função que exiba uma saudação no console.

Então, para poder executá-lo, devemos invocá-lo ou chamá-lo. Que seja algo assim:

```
function cunprimentar() {
 console.log("Olá, seja bem vindo!");
}

cumprimentar()
```

Muito bem, agora você deve modificar a função para que pegue um nome como parâmetro e cumprimente essa pessoa quando a função executar.

Um case estranho

O operador matemático **+** em algumas linguagens e em JS funciona para juntar duas strings de texto ou Strings. Por exemplo, se quisermos juntar uma saudação a um nome, faríamos algo assim:

```
function cumprimentarA(nome) {
 const mensagem = "Olá, seja bem vindo! "
 console.log(mensagem + nome);
}

cumprimentarA("William")
cumprimentarA("Danilo")
```

Como você pode ver, esta função ora não retorna, ora retorna nenhum valor, apenas mostra algo no console. Vamos ver esse retorno?

Funções que retornam valor

Conforme visto anteriormente, uma função é uma máquina, ela insere um valor e retorna um resultado. Em JS, esse exemplo é escrito assim:

```
function multiplicarPorDoisESomarCinco(x) {
 return 2 * x + 5
}
multiplicarPorDoisESomarCinco(1)
multiplicarPorDoisESomarCinco(5)
multiplicarPorDoisESomarCinco(Erick) // Erick não é um número
const Erick= 5
multiplicarPorDoisESomarCinco(Erick) //agora sim
```

Mas, falta alguma coisa aqui. Você vê o resultado no console? O que devemos adicionar?

Claro, o procedimento ou função responsável por exibir valores no console:

```
console.log(multiplicarPorDoisESomarCinco(Erick));
```

Quanto tempo para escrever tudo isso, pode-se salvar em uma variável o que retorna de uma função? Essa variável pode ser passada para o console?

```
const valor = multiplicarPorDoisESomarCinco(Erick)
console.log(valor);
```

Atenção: *Não se esqueça de que o sinal = na programação é usado para atribuir um valor a uma variável, diferente de como é usado em matemática.*

Agora você tem tudo o que precisa para fazer os exercícios nas mesas de trabalho.

Boa sorte!