05

Para saber mais: Tipos de funções

Os parâmetros e o retorno das funções são utilizados de acordo com cada caso específico. Isso significa que nem sempre todas as funções que escrevemos vão precisar de um ou de outro para fazer o que precisam. Abaixo temos mais exemplos para entender melhor algumas situações.

Função sem retorno e sem parâmetro: A função abaixo apenas executa uma instrução, sem a necessidade de disponibilizar o resultado para o restante do código. Neste exemplo escolhemos usar uma string fixa, então não há necessidade de parâmetros.

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

cumprimentar()
```

COPIAR CÓDIGO

Função sem retorno, com parâmetro: similar à anterior, porém agora a função recebe, via parâmetro, o nome da pessoa a ser cumprimentada. Dessa forma é possível reaproveitar a função para que funcione de maneira parecida com o nome de qualquer pessoa (desde que esteja no formato de dado `string`.

```
function cumprimentaPessoa(pessoa){
 console.log(`oi, ${pessoa}!`)
}
```

```
cumprimentaPessoa('Helena')
```

COPIAR CÓDIGO

Função com retorno, sem parâmetro: É possível combinar funções para que cada uma controle apenas uma parte do código e elas trabalhem juntas.

No caso abaixo, a função `cumprimentar()` não precisa receber nenhum parâmetro. Mas logo abaixo vemos que ela está sendo utilizada para montar uma string na função `cumprimentaPessoa(nomePessoa)`. **Isso significa que a string "`Oi gente!`" deve estar disponível para outras partes do programa - ou seja, deve ser retornada com o uso da palavra-chave `return`.**

```
function cumprimentar(){
 return 'Oi gente!'
}

function cumprimentaPessoa(nomePessoa) {
 console.log(`${cumprimentar()} Meu nome é ${nomePessoa}`)
}

cumprimentaPessoa('Paula') // "Oi gente! Meu nome é Paula"
```

COPIAR CÓDIGO

A função `cumprimentaPessoa(nomePessoa)` recebe como parâmetro uma string onde podemos passar qualquer nome no momento em que executamos (ou chamamos) a função. Quando isso acontecer, a função `cumprimentar()` será executada também, e seu valor de retorno - a string `Oi gente!` - vai ocupar o lugar do `${}` onde a função está sendo chamada.

Função com `return` e mais de um parâmetro: Lembrando que as funções podem receber a quantidade de parâmetros necessária, e que o JavaScript identifica os parâmetros pela ordem! Ou seja, no exemplo abaixo o parâmetro `numero1` se refere a `15`, o parâmetro `numero2` se refere a `30` e o parâmetro `numero3` se refere a `45`. Somos nós, que estamos desenvolvendo o código, que damos os nomes aos parâmetros de acordo com o dado que a função espera receber - no caso, números.

```
function operacaoMatematica(numero1, numero2, numero3) {
 return numero1 + numero2 + numero3
}

operacaoMatematica(15, 30, 45) // 90
```

COPIAR CÓDIGO

Parâmetros x argumentos: Na prática se referem ao mesmo tipo de dado; algumas documentações se referem a *parâmetros* no momento em que a função é definida (no caso, `numero1`, `numero2`, etc) e *argumentos* como os dados que utilizamos para executar a função (ou seja, `30`, `45`, etc).

Ainda há muito o que estudar no tema de funções, então pratique bastante pois parâmetros e retorno são conceitos essenciais.