Deixando as funções dinâmicas:

Até esse ponto, aprendemos a criar funções de uma maneira super bacana, sempre se atentando ao uso da palavra reservada seguida do nome. Mas, nossas funções ainda não estão legais, porque ela sempre faz a mesma coisa.

Você deve estar se perguntando: “não era esse o objetivo?” E a resposta é sim, porém, podemos ter alguns problemas com isso. Imagine a seguinte situação:

Estamos criando uma função chamada somar, em que iremos pegar dois números e simplesmente somá-los. Talvez você pense no seguinte:

```
function somar(){
      10+20
}
```

De início, pode fazer sentido. Afinal, estamos cumprindo o requisito de somar dois números. Mas, por que exatamente 10 e 20? E se quiséssemos somar 60 + 40? Será que teríamos que criar uma nova função?

Bom, a boa notícia é que não precisamos criar uma nova função, e sim utilizar um recurso nativo das funções, chamado de **parâmetros.**

Eles vão nos ajudar a resolver esse problema de uma forma bem simples. Ao invés de trabalharmos com os valores fixos, vamos trabalhar com informações dinâmicas. Na prática, iremos dizer que para executar essa função, precisamos passar dois números e apenas somá-los. Para isso, iremos definir que a função irá receber dois parâmetros:

```
function somar(numeroA, numeroB){

}
```

Esses parâmetros vão se comportar como variáveis dentro da nossa função. Eles terão os valores que forem passados no momento da execução da função, e então poderemos fazer o que quisermos com eles como, por exemplo, somá-los.

```
function somar(numeroA, numeroB){
    numeroA + numeroB
}
```

Dessa forma, independente do valor passado, iremos somá-lo. A ideia é garantir uma melhor performance e aplicabilidade da nossa função!

Isso pode gerar uma outra dúvida, já que precisamos receber o valor calculado, ou seja, o resultado da soma. Podemos até pensar em imprimir o valor com console.log(), mas não poderemos usar o resultado para fazer outros cálculos e afins.

Então, surge uma outra palavra reservada que irá nos ajudar: **return** ou em português, **retornar.** Como o nome já diz, ela irá nos retornar um valor, seja ele número, string, array e afins.

```
function somar(numeroA, numeroB){
    return numeroA + numeroB
}
```

Nesse caso, a função que estamos usando irá retornar o resultado da soma. Com isso, deixamos as coisas bem completas: criamos uma função que recebe dois valores, e que irá retornar a soma entre eles. Legal, né?

Vamos só esclarecer alguns pontos importantes:

Em uma função, não é obrigatório o uso de return. Você precisa entender o que sua função precisa fazer, ou seja, se ela irá apenas realizar uma ação, como por exemplo exibir um console, ou se ela irá devolver algo para ser usado em algum momento do nosso código.

E, por último, **o return é SEMPRE a última linha de uma função**. Sempre que o computador interpretar essa linha, ele entende que a função acabou ali. Qualquer código após o return de uma função será desconsiderado:

```
function dizerOla(){
      return "olá"
 // o console logo a baixo não será executado devido ao return acima!
      console.log('Oi também!')
}
```

Mas, como podemos pegar o retorno da função? Como executar uma função?

Opa...Calma. Iremos responder a todas essas perguntas à frente. Por agora, vamos focar mais uma vez na criação das funções em conjunto dos parâmetros, e retornando os dados que queremos. Que tal fazermos alguns exercícios?