ntander Coders 2024.1 | Front-End | #1178

arametrização de funções

Isaias Soares
Id: 1178008

Parametrização de funções

Observe o seguinte exemplo:

Exemplo: Somar

```
function somar() { console.log(`1 + 2 = ${1 + 2}`); }

somar() // 1 + 2 = 3
```

Essa é uma função de somar, ela produz no console o resultado da soma de 1 com 2. Sendo assim, sempre que quisermos somar 1 e 2 podemos invocá-la. Super útil, não é!?

Nem tanto...

Para tornar funções mais reutilizáveis, em diversos casos em que a lógica é a mesma mas os valores são diversos (como em somas) adicionamos parâmetros às funções, permitindo que elas usem valores escolhidos apenas na hora de invocá-las.

Exemplo: Somar com parâmetros

```
function somar(num1, num2) {
  console.log(`${num1} + ${num2} = ${num1 + num2}`);
}

somar(100, 200) // 100 + 200 = 300
```

Criar parâmetros é fácil, basta colocar nomes (como variáveis) entre parênteses na hora de criar sua função. Você pode colocar quantos precisar desde que separados por vírgula.

Quando sua função for chamada (invocada) basta passar os valores dos parâmetros entre parênteses que eles serão enviados para as "variáveis" que criamos na função.

No exemplo acima, `somar()` é invocado passando os valores 100 e 200 como parâmetros, como a ordem dos parâmetros é respeitada, `num1` terá o valor 100 e `num2` terá o valor 200.

No JavaScript, nada impede de que uma função seja invocada com parâmetros faltando, ou de tipos incorretos, então tome cuidado quando estiver criando funções para outras pessoas utilizarem, pois os parâmetros podem vir inconsistentes. É necessário fazer validação e gerar erros caso os parâmetros estejam inconsistentes.

Funções em JavaScript são naturalmente polimórficas, isso quer dizer que elas permitem que em seus parâmetros aceitem qualquer tipo de dados. Isso quer dizer que nossa função de soma acima funcionaria com float, com int, e até com strings, porque todos esses tipos têm o operador `+`.

Veja:

```
somar(10.5,20.7);              // 10.5 + 20.7 = 31.2
somar(10, 20);                 // 10 + 20 = 30
somar("isso é ", "um teste!"); // isso é + um teste! = isso é um teste!
```

No entanto, isso também quer dizer que ela pode ser chamada como tipos que não possuem o operador `+` e isso produzirá um erro ou ela irá se comportar de forma imprevisível.

Referências e Materiais Complementares

- Parâmetros de função
- Como o Javascript funciona: entendendo as funções e suas formas de uso
- Entendendo os parâmetros predefinidos em JavaScript

Próximo Tópico