Lógica de Programação

Aula 07
Funções

Prof. Anderson Passos de Aragão

Agenda

- Revisão
- Funções

Revisão

- Arrays

Funções

Introdução

As funções em JavaScript são **blocos de código reutilizáveis** que podem ser chamados para executar uma **tarefa específica**

Elas são essenciais para organizar e modularizar o código

Para usar uma função, você deve defini-la em algum lugar no escopo do qual você quiser chamá-la

Declaração de Funções

Existem duas maneiras de declarar funções: expressões de função e declarações de função

Veremos as duas formas em detalhes

Declaração de Função

A definição da função (também chamada de declaração de função) consiste no uso da palavra chave **function**, seguida por:

- Nome da Função.
- Lista de argumentos para a função, entre parênteses e separados por vírgulas.
- Declarações JavaScript que definem a função, entre chaves { }.

Declaração de Função

```
1  function quadrado(numero){
2      return numero*numero
3  }
4
5  console.log(quadrado(2))
```

A função **quadrado** recebe um argumento chamado **numero**

A função consiste em uma instrução que indica para retornar o argumento da função (isto é, numero) multiplicado por si mesmo

A declaração **return** especifica o valor retornado pela função

O resultado da execução deste programa é o número **4**

Declaração de Função

Parâmetros primitivos (como um número) são passados para as funções por valor

O valor é passado para a função, mas se a **função altera o valor do parâmetro**, esta mudança não reflete globalmente ou na função chamada.

Se você passar um objeto (ou seja, um valor não primitivo, tal como Array definido por você) como um parâmetro e a função alterar as propriedades do objeto, essa mudança é visível fora da função

Declaração de Função

Parâmetros primitivos (como um número) são passados para as funções por valor

O valor é passado para a função, mas se a **função altera o valor do parâmetro**, esta mudança não reflete globalmente ou na função chamada.

Se você passar um objeto (ou seja, um valor não primitivo, tal como Array definido por você) como um parâmetro e a função alterar as propriedades do objeto, essa mudança é visível fora da função

Expressão de Função

Embora a declaração de função acima seja sintaticamente uma declaração, funções também podem ser criadas por uma **expressão de função**

Tal função pode ser anônima (ela não tem que ter um nome)

```
1 let quadrado = function (numero){
2     return numero*numero
3 };
4 console.log(quadrado(4))
```

Expressão de Função

No entanto, um nome pode ser fornecido com uma expressão de função e pode ser utilizado no interior da função para se referir a si mesma

Esse processo é chamado de **recursão**

```
let fatorial = function fat(n){
  if(n == 1 || n == 0){
      return 1
  } else {
      return n*fat(n-1)
  }
};

console.log(fatorial(4))
```

Expressão de Função

No entanto, um nome pode ser fornecido com uma expressão de função e pode ser utilizado no interior da função para se referir a si mesma

Esse processo é chamado de **recursão**

```
let fatorial = function fat(n){
  if(n == 1 || n == 0){
    return 1
  } else {
    return n*fat(n-1)
  }
};

console.log(fatorial(4))
```

Chamando Funções

Após declarar uma função, você pode chamá-la para executar o código dentro dela.

Para chamar uma função você coloca o nome dela e o(s) parâmetro(s) necessários.

```
1  function quadrado(numero){
2      return numero*numero
3  }
4
5  console.log(quadrado(2))
```

Parâmetros e Argumentos

Funções podem receber parâmetros, que são variáveis usadas na definição da função.

numero é um parâmetro da função quadrado

```
1  function quadrado(numero){
2      return numero*numero
3  }
4
5  console.log(quadrado(2))
```

Retorno de Funções

As variáveis definidas no interior de uma função não podem ser acessadas de nenhum lugar fora da função, porque a variável está definida apenas no escopo da função.

```
function soma(n1, n2){
    n1 = n1*2
    n2 = n2*2
    console.log(`Escopo Local n1 = ${n1} e n2 = ${n2}`)
    return n1 + n2
}

let n1 = parseInt(prompt("Digite um número: "))
let n2 = parseInt(prompt("Digite um número: "))
console.log(soma(n1, n2))
console.log(`Escopo Global n1 = ${n1} e n2 = ${n2}`)
```

Retorno de Funções

As variáveis definidas no interior de uma função não podem ser acessadas de nenhum lugar fora da função, porque a variável está definida apenas no escopo da função

No entanto, uma função pode acessar todas variáveis e funções definida fora do escopo onde ela está definida

Em outras palavras, a função definida no escopo global pode acessar todas as variáveis definidas no escopo global.

Retorno de Funções

```
1  function soma(n1, n2){
2      n1 = n1*2
3      n2 = n2*2
4      console.log(`Escopo Local n1 = ${n1} e n2 = ${n2}`)
5      return n1 + n2
6  }
7
8  let n1 = parseInt(prompt("Digite um número: "))
9  let n2 = parseInt(prompt("Digite um número: "))
10 console.log(soma(n1, n2))
11 console.log(`Escopo Global n1 = ${n1} e n2 = ${n2}`)
```

Encerramento

- Revisão
- Exercício Complementar
- Próxima Aula