Funções

De modo geral, **função** é um "subprograma" que pode ser chamado por código externo (ou interno no caso de recursão) à função.

Assim como o programa em si, uma função é composta por uma sequência de instruções chamada corpo da função.

Valores podem ser passados para uma função e ela vai retornar um valor.

Em JavaScript, funções são objetos de primeira classe, pois elas podem ter propriedades e métodos como qualquer outro objeto.

```
function nomeDaFuncao(){
        //o que a função faz (corpo da função)
}
```

Funções declaradas

O jeito mais básico de definir funções em JavaScript é através da função declarada (function declaration), toda função de declaração começa com a palavra reservada e obrigatória function, seguida pelo nome da função (também obrigatório) e uma lista de parâmetros (opcionais) separados por vírgula e encapsulados em parenteses (obrigatórios), o último passo é definir as chaves (obrigatórias) que será o corpo da função.

```
// criando a função
function escrever(){
   let idade = 18
   let nome = 'fulano'
   console.log(`Meu nome é ${nome} e eu tenho ${idade} de idade`);
}

// Chamando a função
escrever()
```

Funções declaradas com parâmetros

```
// criando a função
function escrever(idade, nome){
   console.log(`Meu nome é ${nome} e eu tenho ${idade} de idade`);
}
```

```
let idade = 18
let nome = 'fulano'
// Chamando a função e passando parâmetros
escrever(idade, nome)
```

Expressões de função

A function expression (expressão de função) é muito parecida com a function declaration, a diferença é que uma função de expressão pode ser lidada como uma qualquer expressão em JavaScript, devendo ser atribuída a uma variável.

```
let nomeDaFuncao =  function (){
        //o que a função faz (corpo da função)
}
```

Repare que é bem parecido com as funções de declaração, uma das sútis diferenças é que ela está sendo atribuída para uma variável, onde não definimos o nome da função e sim o nome da variável que irá referenciar a mesma.

```
// criando a função
let escrever  =  function(){
   let idade = 18
   let nome = 'fulano'
   console.log(`Meu nome é ${nome} e eu tenho ${idade} de idade`);
}

// Chamando a função
escrever()
```

Expressões de função com parâmetro

```
// criando a função
let escrever  =  function(nome, idade){
   console.log(`Meu nome é ${nome} e eu tenho ${idade} de idade`);
}

// Chamando a função
let idade = 18
let nome = 'fulano'
escrever(idade, nome)
```

Arrow Functions

Arrow functions são simplificações para as functions expression..

```
let escrever = () => {
   let idade = 18
   let nome = 'fulano'
   console.log(`Meu nome é ${nome} e eu tenho ${idade} de idade`);
}

escrever()
```

Arrow Functions com parâmetros

```
let escrever = (idade, nome) => {
   console.log(`Meu nome é ${nome} e eu tenho ${idade} de idade`);
}

let idade = 18
let nome = 'fulano'
escrever(idade, nome)
```

Capturando dados de um formulário

HTML

Criando o formulário

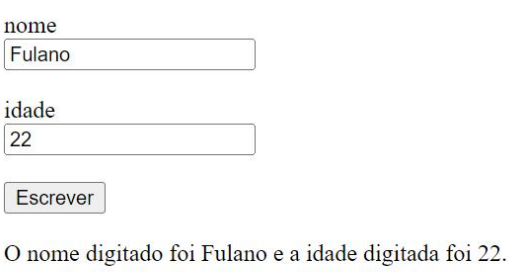
```
<label for="nome">nome</label>
<input type="text" id="nome">

<label for="idae">idade</label>
<input type="text" id="idade">

<button id="btn">Escrever</button>

<div id="app"></div>
```

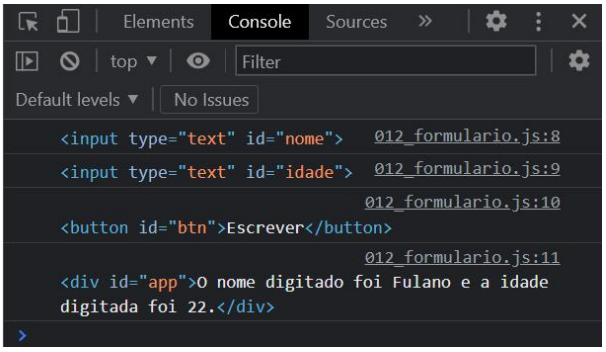
JavaScript

Capturando os dados e exibindo os dados na tela.

```
// capturando os dados
let nome = document.getElementById('nome')
let idade = document.getElementById('idade')
let btn = document.getElementById('btn')
let div = document.getElementById('app')

// criando a função para exibir os dados
function escrever(){
   // exibindo os elementos no console
   console.log(nome);
   console.log(idade);
   console.log(btn);
   console.log(app);

   // inserindo os dados formatados no componente DIV
   // para pegar o valor devemos utilizar '.value'
   div.innerHTML = `O nome digitado foi ${nome.value} e a idade digitada foi
${idade.value}.`

}

// vinculando o evento de clicar no botão para chamar a função escrever
btn.addEventListener('click', escrever)
```

Testando a aplicação





Estruturas condicionais

As **estruturas condicionais** são elementos fundamentais na programação, pois elas permitem que um programa tome decisões com base em condições específicas. Elas permitem que você controle o fluxo de execução do código, determinando se certas instruções devem ser executadas ou não, dependendo de sua condição (verdadeira ou falsa).

As estruturas condicionais permitem que você possa verificar se uma determinada expressão lógica (condição) é verdadeira ou não.

Se a expressão for verdadeira ela executa um determinado trecho do código e em caso contrário poderá ser executado outro trecho de código ou simplesmente não executar nenhum trecho de código.

Uma expressão condicional sempre irá retornar um valor booleano, em outras palavras ela sempre irá retornar um valor **verdadeiro** ou **falso**.

As estruturas condicionais são importantes por várias razões:

1. **Tomada de Decisões**: As estruturas condicionais permitem que os programas tomem decisões lógicas com base em dados ou entradas do usuário. Isso é essencial para criar programas que se adaptem dinamicamente às circunstâncias.

2. **Controle de Fluxo**: Elas fornecem controle de fluxo, permitindo que você especifique diferentes caminhos de execução do programa com base nas condições. Isso é útil para criar programas que respondam de maneira apropriada a diferentes cenários.

3. **Eficiência**: As estruturas condicionais também podem aumentar a eficiência do código, evitando a execução de instruções desnecessárias. Se uma condição não for satisfeita, as instruções relacionadas a ela podem ser ignoradas.

4. **Personalização**: Com as estruturas condicionais, você pode personalizar a interação do seu programa com o usuário ou com os dados, tornando-o mais flexível e útil.

Existem vários tipos de estruturas condicionais em programação, incluindo:

- **IF**: O "if" é a estrutura condicional mais básica, que permite que um bloco de código seja executado somente se uma condição for verdadeira.

- **ELSE**: O "else" é obrigatóriamente usado em conjunto com o "if" e permite a execução de um bloco de código alternativo caso a condição do "if" seja falsa.

- **ELSE IF (ou ELIF)**: O "else if" (ou "elif" em algumas linguagens) é usado para verificar múltiplas condições sequencialmente e executar o bloco de código associado à primeira condição verdadeira encontrada.

- **Switch/Case (ou Seleção por Caso)**: Esta estrutura é usada para lidar com múltiplas condições diferentes e direcionar a execução do código para diferentes blocos de instruções com base no valor de uma variável ou expressão.

Em resumo, as estruturas condicionais são fundamentais para criar programas que podem tomar decisões lógicas e se adaptar a diferentes situações, tornando-os mais

poderosos e flexíveis. Elas desempenham um papel crucial na lógica de programação e são uma habilidade essencial para qualquer desenvolvedor de software.

Operador	Significado
==	Igual a
!=	Diferente de
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a
===	Idêntico
!==	Não idêntico

Estrutura condicional IF..., IF... ELSE e IF... ELSE IF... ELSE

Estrutura Condicional "IF"

A estrutura condicional "**if**" é usada para executar um bloco de código apenas se uma determinada condição for verdadeira. Aqui está a sintaxe básica:

```
if (condicao) {
        // Código a ser executado se a condição for verdadeira
}
```

Exemplo:

```
let idade = 20;
if (idade >= 18) {
  console.log("Você é maior de idade.");
}
```

Neste exemplo, o código dentro do bloco "if" só será executado se a variável "idade" for maior ou igual a 18.

Estrutura Condicional "IF/ELSE"

A estrutura condicional "**if/else**" é usada quando você deseja executar um bloco de código se uma condição for verdadeira e outro bloco de código se a condição for falsa. Aqui está a sintaxe básica:

```
if (condicao) {
        // Código a ser executado se a condição for verdadeira
} else {
        // Código a ser executado se a condição for falsa
}
```

Exemplo:

```
let idade = 15;
if (idade >= 18) {
  console.log("Você é maior de idade.");
} else {
  console.log("Você é menor de idade.");
}
```

Neste exemplo, se a idade for maior ou igual a 18, a mensagem "Você é maior de idade." será exibida. Caso contrário, a mensagem "Você é menor de idade." será exibida.

IF... ELSE IF... ELSE

A estrutura condicional "if...else if...else" é usada quando você precisa avaliar várias condições sequencialmente e executar diferentes blocos de código com base na primeira condição verdadeira encontrada. Isso permite criar uma cadeia de verificações condicionais mais complexas. Aqui está a sintaxe básica:

```
if (condicao1) {
        // Código a ser executado se a condição1 for verdadeira
} else if (condicao2) {
        // Código a ser executado se a condição2 for verdadeira
} else {
        // Código a ser executado se nenhuma das condições anteriores for verdadeira
}
```

Vamos ver um exemplo em JavaScript para ilustrar como isso funciona:

```
let diaSemana = "quarta";

if (diaSemana === "segunda") {
  console.log("É segunda-feira.");
} else if (diaSemana === "terça") {
  console.log("É terça-feira.");
} else if (diaSemana === "quarta") {
  console.log("É quarta-feira.");
} else {
  console.log("É um dia diferente da semana.");
}
```

Neste exemplo, a variável diaSemana é avaliada em várias condições sequencialmente. Se diaSemana for igual a "segunda", será exibida a mensagem "É segunda-feira." Se for igual a "terça", a mensagem será "É terça-feira." Se for igual a "quarta", a mensagem será "É quarta-feira." Se nenhuma das condições anteriores for verdadeira, a mensagem "É um dia diferente da semana." será exibida.

Esta estrutura condicional é útil quando você tem várias condições para verificar e deseja executar um bloco de código correspondente à primeira condição verdadeira encontrada. Tenha em mente que o código dentro de um bloco "else if" só será

executado se a condição associada ao "if" ou ao "else if" correspondente for verdadeira. Se nenhuma das condições for verdadeira, o bloco "else" será executado (se presente).

Operador Ternário

O **operador ternário** em JavaScript é uma forma concisa de escrever uma estrutura condicional "**if...else**" em uma única linha. Ele é frequentemente usado para atribuir um valor a uma variável com base em uma condição. A sintaxe básica do operador ternário é a seguinte:

```
condicao ? valorSeVerdadeiro : valorSeFalso
```

Onde:

- **condicao**: A expressão condicional que será avaliada. Se esta condição for verdadeira, o valor antes do dois pontos (:) será retornado. Caso contrário, o valor após o dois pontos (:) será retornado.
- **valorSeVerdadeiro**: O valor a ser retornado se a condição for verdadeira.
- **valorSeFalso**: O valor a ser retornado se a condição for falsa.

Aqui está um exemplo simples:

```
let idade = 20;
let mensagem = idade >= 18 ? "Maior de idade" : "Menor de idade";

console.log(mensagem); // Isso irá imprimir "Maior de idade"
```

Neste exemplo, a condição **idade >= 18** é avaliada. Se for verdadeira, a mensagem "Maior de idade" será atribuída à variável mensagem; caso contrário, a mensagem "Menor de idade" será atribuída. Como idade é 20, a primeira condição é verdadeira e, portanto, "Maior de idade" é atribuído à variável mensagem.

Outros exemplos:

```
let nota = 5.00;
let texto = (nota >= 6.00) ? "Aprovado" : "Reprovado";
console.log(texto);


nota = 9.99;
console.log((nota >= 6.00) ? "Aprovado" : "Reprovado");
```

O operador ternário é útil quando você deseja atribuir um valor com base em uma condição de maneira sucinta e legível. No entanto, ele deve ser usado com moderação

para manter o código legível, já que pode se tornar confuso quando usado em expressões muito complexas.

Operadores lógicos

Operador lógico	Significado
&& (and)	Se duas expressões condicionais forem verdadeiras, o resultado é verdadeiro
|| (or)	Se qualquer expressão condicional é verdadeira, o resultado é verdadeiro
! (not)	Se a expressão condicional for falsa, o resultado é verdadeiro. Se a expressão condicional for verdadeira, o resultado é falso

Em JavaScript, os operadores lógicos **&&** (E lógico), **||** (OU lógico) e **!** (NÃO lógico) são usados para realizar operações lógicas em valores booleanos (verdadeiro ou falso). Eles são frequentemente usados em estruturas condicionais para criar expressões lógicas mais complexas. Vamos explicar cada um deles:

1. **Operador && (E lógico)**:

O operador **&&** realiza uma operação de **E lógico** entre dois valores booleanos. Ele retorna **true** se ambos os operandos forem verdadeiros e **false** se pelo menos um deles for falso.

Exemplo:

```
let a = true;
let b = false;

let resultado = a && b; // resultado é igual a false
```

Neste exemplo, **resultado** será *false* porque ambos **a** e b precisam ser verdadeiros para que a expressão seja verdadeira.

Outro exemplo com o operador lógico &&:

Suponha que você queira verificar se uma pessoa é elegível para votar em uma eleição com base em sua idade e cidadania. Para votar, a pessoa deve ser maior de idade (18 anos ou mais) e ser cidadã do país. Você pode usar o operador && para verificar ambas as condições:

```
let idade = 20;
let eCidadao = true;

if (idade >= 18 && eCidadao == true) {
```

```
  console.log("Você é elegível para votar.");
} else {
  console.log("Você não é elegível para votar.");
}
```

Neste exemplo, o código dentro do bloco "if" só será executado se ambas as condições, idade >= 18 e eCidadao == true, forem verdadeiras.

2. Operador || (OU lógico):

O operador || realiza uma operação de OU lógico entre dois valores booleanos. Ele retorna true se pelo menos um dos operandos for verdadeiro e false somente se ambos os operandos forem falsos.

Exemplo:

```
let a = true;
let b = false;

let resultado = a || b; // resultado é igual a true
```

Neste exemplo, resultado será true porque pelo menos um dos operandos (a) é verdadeiro.

Outro exemplo com o operador lógico ||:

Agora, suponha que você deseje verificar se um usuário tem acesso a um conteúdo premium com base em sua assinatura mensal ativa ou em uma assinatura anual ativa. Se qualquer uma das condições for verdadeira, o usuário terá acesso ao conteúdo premium:

```
let assinaturaMensalAtiva = false;
let assinaturaAnualAtiva = true;

if (assinaturaMensalAtiva || assinaturaAnualAtiva) {
  console.log("Você tem acesso ao conteúdo premium.");
} else {
  console.log("Você não tem acesso ao conteúdo premium.");
}
```

Neste exemplo, o código dentro do bloco "if" será executado se pelo menos uma das condições, assinaturaMensalAtiva ou assinaturaAnualAtiva, for verdadeira.

3. Operador ! (NÃO lógico):

O operador ! (NÃO lógico) é usado para inverter o valor de uma expressão booleana. Ou seja, ele retorna true se a expressão for falsa e false se a expressão for verdadeira.

Exemplo:

```
let a = true;
let resultado = !a; // resultado é igual a false
```

Neste exemplo, resultado será **false** porque **!a** inverte o valor de 'a', que era **true**.

Outro exemplo com o operador lógico !:

Imagine que você deseja verificar se um usuário não está banido de um fórum. Você pode usar o operador '**!**' para inverter o valor de uma variável que indica se o usuário está banido:

```
let usuarioBanido = false;

if (!usuarioBanido) {
  console.log("Você pode postar no fórum.");
} else {
  console.log("Você está banido do fórum.");
}
```

Neste exemplo, o operador '**!**' é usado para inverter o valor de **usuarioBanido**, para que o código dentro do bloco "**if**" seja executado quando o usuário não estiver banido.

Os operadores lógicos são fundamentais para criar condições mais complexas em estruturas condicionais, onde você pode combinar várias expressões booleanas para tomar decisões com base em critérios mais elaborados. Eles também são úteis em outras situações em que a lógica booleana é necessária, como validação de formulários, controle de loops e etc.

Exercício 44: Cálculo de Idade em Dias

Você deve calcular a idade de uma pessoa em dias. Insira sua idade em anos e, em seguida, calcule e exiba a idade em dias. Lembrando que um ano tem 365 dias. (Exiba os dados no browser e no console).

Exercício 45:

Antes do racionamento de energia ser decretado, quase ninguém falava em quilowatts, mas agora, todos incorporaram essa palavra em seu vocabulário.

Sabendo-se que 100 quilowatts de energia custa um sétimo do salário mínimo, faça uma aplicação que receba o valor do salário mínimo e a quantidade de quilowatts gasta por uma residência.

Calcule e exiba no browser:

- O valor em reais de cada quilowatt.
- O valor em reais a ser pago.
- O novo valor a ser pago por essa residência com um desconto de 10%.

Exercício 46:

Crie uma aplicação que receba a temperatura em graus Celsius e a exiba convertida em graus Fahrenheit.

A fórmula de conversão é: F = C*(9.0/5.0)+32.0, sendo F a temperatura em Fahrenheit e C a temperatura em Celsius. (Exiba os dados no browser e no console).

Exercício 47:

Crie uma aplicação que receba o salário de um funcionário e exiba o valor deste salário, calcular e mostrar seu novo salário, sabendo que ele recebeu um aumento de 25%. (Exiba os dados no browser e no console).

Exercício 48:

Crie uma aplicação que receba a altura do degrau de uma escada (cm) e a altura que o usuário deseja alcançar subindo a escada (cm). Calcule e mostre quantos degraus o usuário deverá subir para atingir seu objetivo, sem se preocupar com a altura do usuário. (Exiba os dados no browser e no console).

Exercício 49: IMC

Crie um programa que calcule o Índice de Massa Corporal (IMC) de uma pessoa. O programa deve receber o peso em quilogramas e a altura em metros. O programa deve exibir o IMC calculado. (Exiba os dados no browser e no console, os valores devem ser obrigatoriamente maiores que zero).

A fórmula para calcular o IMC é:

IMC = peso (em quilogramas) / (altura (em metros) * altura (em metros))

Exercício 50: Calculando o Preço Total de Compras

Crie um programa para calcular o preço total de compras em uma loja. O programa deve receber o preço de três produtos diferentes. O programa deve somar os preços e exibir o total a ser pago sem desconto, com desconto de 10%, desconto de 20%, desconto de 30% e desconto de 50%. (Exiba os dados no browser e no console).

Exercício 51: Calculando a Média de Notas

Crie um programa para calcular a média aritmética de cinco notas. O programa deve receber as cinco notas, deve calcular a média aritmética e exibir o resultado no browser.

Exercício 52: Cálculo de Gorjeta

Faça um programa que calcule a gorjeta a ser deixada em um restaurante. O programa deve receber o valor da conta e a porcentagem de gorjeta desejada. O programa deve calcular e exibir no browser o valor da gorjeta e o total a ser pago.