JavaScript

Gustavo Marino Botta

HTML, CSS e JavaScript

- HTML para definir o conteúdo das páginas da web
- CSS para especificar o layout das páginas da web
- JavaScript para programar o comportamento das páginas da web

O que JavaScript?

- JavaScript é uma linguagem de programação que adiciona interatividade em sua página web, por exemplo, jogos, respostas a eventos como um botão pressionado, estilização dinâmica, animações, etc.

Você sabia?

- JavaScript e Java são completamente diferentes.
- JavaScript foi inventado pelo Brendan Eich em 1995, e tornou-se um padrão ECMA em 1997.
- ECMAScript é uma especificação da linguagem e o JavaScript implementa essa especificação.

Para aprender...

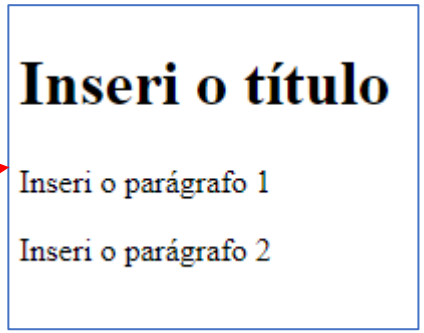
- O site
  - https://www.w3schools.com/js/default.asp mantém um guia de referência completo da linguagem JavaScript e está em constante atualização. Além disso, apresenta exemplos das propriedades e métodos.

- Vale a pena conferir!

Onde colocar os códigos JavaScript?

Dentro da tag HEAD

```
<head>
   <script>
      window.onload = funcao01;
      function funcao01() {
         document.getElementById("id02").innerHTML = "Inseri o parágrafo 1"
      }
   </script>
</head>
<body>
   <h1 id="id01"></h1>
   <p id="id02"></p>
   <p id="id03"></p>
   <script>
      document.getElementById("id01").innerHTML = "Inseri o título";
   </script>
   <script src="exemplo.js"></script>
</body>
```



No corpo da página

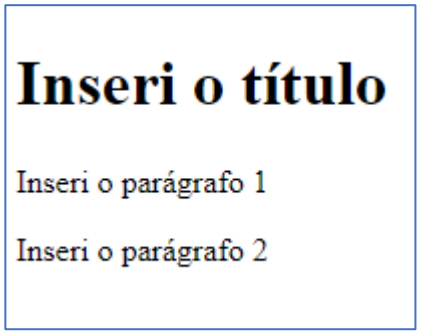
No arquivo externo

```
// Arquivo externo exemplo.js
document.getElementById("id03").innerHTML = "Inseri o parágrafo 2"
```

Colocar scripts na parte inferior do elemento <body> melhora a velocidade de exibição, porque a interpretação do script torna a exibição mais lenta.

Onde colocar os códigos JavaScript?

```
<head>
   <script>
      window.onload = funcao01;
      function funcao01() {
         document.getElementById("id02").innerHTML = "Inseri o parágrafo 1"
      }
   </script>
</head>
<body>
   <h1 id="id01"></h1>
   <p id="id02"></p>
   <p id="id03"></p>
   <script>
      document.getElementById("id01").innerHTML = "Inseri o título";
   </script>
   <script src="exemplo.js"></script>
</body>
```




```
// Arquivo externo exemplo.js
document.getElementById("id03").innerHTML = "Inseri o parágrafo 2"
```

Script externo

- Você pode colocar uma referência de script externa em <head> ou <body> como desejar.

- O script se comportará como se estivesse localizado exatamente onde a tag <script> está localizada.

- Vantagens do JavaScript externo
  - Separa HTML e código
  - Torna o HTML e o JavaScript mais fáceis de ler e manter
  - Arquivos JavaScript em cache podem acelerar o carregamento da página
  - Para adicionar vários arquivos de script a uma página - use várias tags de script:
    - <script src="myScript1.js"></script>
    - <script src="myScript2.js"></script>

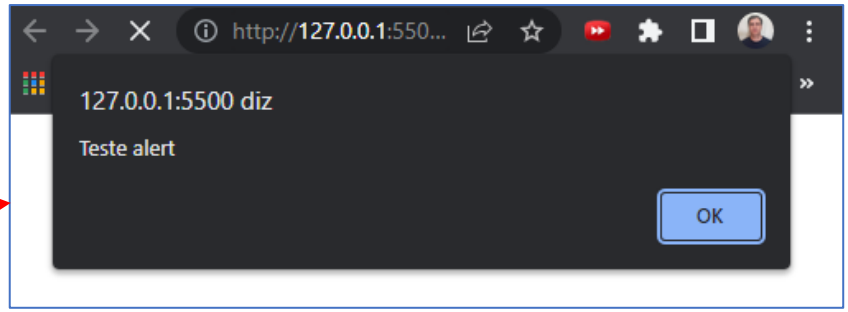
Exibindo dados de diferentes maneiras

```
<h1 id="id01"></h1>
<script>
    document.getElementById("id01").innerHTML =
"Teste innerHTML";
    document.write("Teste write");
    window.alert("Teste alert");
    console.log("Teste console");
</script>
```

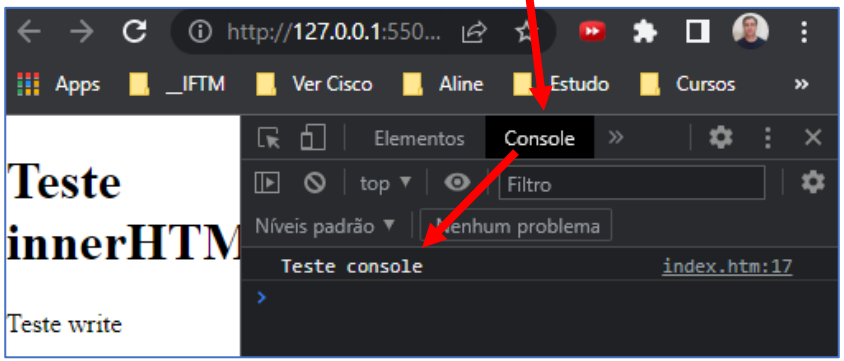
- Escrevendo em um elemento HTML, usando innerHTML.
- Escrevendo na saída HTML usando document.write().
  - O uso de document.write() após o carregamento de um documento HTML excluirá todo o HTML existente
  - O método document.write() deve ser usado apenas para teste.
- Escrevendo em uma caixa de alerta, usando window.alert().
- Escrevendo no console do navegador, usando console.log().

Exibindo dados de diferentes maneiras

```
<h1 id="id01"></h1>
<script>
    document.getElementById("id01").innerHTML =
"Teste innerHTML";
    document.write("Teste write");
    window.alert("Teste alert");
    console.log("Teste console");
</script>
```

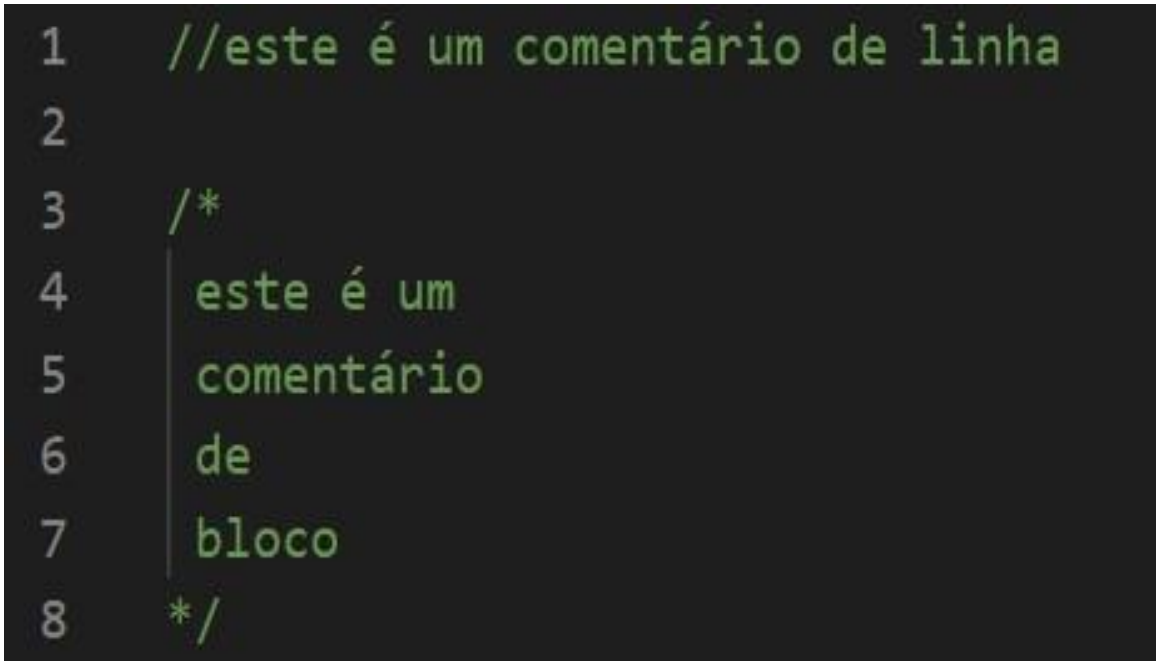


Tecla F12



Comentários

- Pode ser feito de linha ou de bloco.



Variáveis

- São conteineres para armazenar valores de dados.
- São fracamente tipadas: não é necessário declarar o seu tipo.
- Devem ter identificadores únicos.
  - Pode conter letras, dígitos, underscore (_) e sinal de cifrão ($)
  - Pode começar com letras, _ e $
  - Os nomes são case sensitive
  - Palavras reservadas não podem ser utilizadas
  - Hifens não são permitidos em JavaScript. Eles são reservados para subtrações.
- Pode ser declarada utilizando: **var**, **let** ou **const**

Variáveis

- Declarações

```
var numero = 10;

let numero = 10;

const numero = 10;
```

- Diferenças entre **var**, **let** e **const**
  - **var** – escopo de função
  - **let** – escopo de bloco e mutável
    - não podem ser redeclaradas.
    - devem ser declaradas antes do uso.
  - **const** – escopo de bloco e imutável
    - não podem ser redeclaradas.
    - não podem ser reatribuídas.
    - devem receber um valor quando são declaradas.

Variáveis

- Diferença entre **var** e **let**:

```
1   if(true){
2     var x = 5;
3   }
4   console.log(x);
```

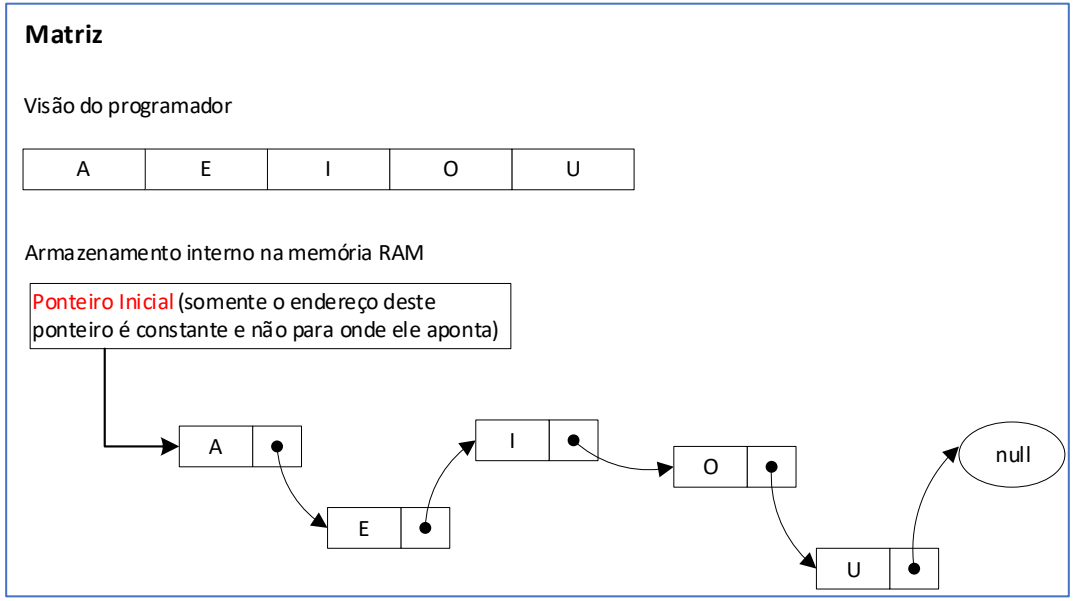
```
1   if(true){
2     let x = 5;
3   }
4   console.log(x);
```

Variáveis – outras características

- Uma variável declarada sem valor terá o valor undefined.
- Strings são escritas entre aspas duplas ou simples. Os números são escritos sem aspas.
- Se você colocar um número entre aspas, ele será tratado como uma string de texto.
- `let x = "5" + 2 + 3;  // resultará em 523`
  - Se você colocar um número entre aspas, o restante dos números será tratado como strings e concatenado.

Objetos e matrizes constantes

- A palavra-chave const é um pouco enganosa.
- Não define um valor constante. Ele define uma referência constante a um valor.
- NÃO pode:
  - Reatribuir um valor constante
  - Reatribuir uma matriz constante
  - Reatribuir um objeto constante
- Pode:
  - Alterar os elementos da matriz constante
  - Alterar as propriedades do objeto constante



Objeto é semelhante a matriz mas é composto por propriedades (par de nome:valor).

Operadores

- Operador unário

operador **<operando> ou <operando>** ***operador***

X++

- Operador binário:

<operando1> ***operador*** **<operando2>**

X + Y

Existe também um operador ternário que veremos adiante.

Operadores Aritméticos

Operador	Descrição	Exemplo
Módulo (%)	Operador binário. Retorna o inteiro restante da divisão dos dois operandos	12 % 5 retorna 2.
Incremento (++)	Operador unário. Adiciona uma unidade ao seu operando.	x++, se x é 3, então retorna 4.
Decremento (--)	Operador unário. Subtrai uma unidade de seu operando.	x--, se x é 3, então retorna 2.
Negação (-)	Operador unário. Retorna a negação de seu operando.	Se x é 3, então –x retorna -3.
Adição (+)	Operador unário. Tenta converter o operando em um número sempre que possível	+”3” retorna 3. +true retorna 1.
Operador de exponenciação (**)	Calcula base^expoente	2**3 retorna 8.

Operadores de Atribuição

Nome	Operador encurtado	Significado
Atribuição	x = y	x = y
Atribuição de adição (se usado com string faz a função de concatenação)	x += y	x = x + y
Atribuição de subtração	x -= y	x = x - y
Atribuição de multiplicação	x *= y	x = x * y
Atribuição de divisão	x /= y	x = x / y
Atribuição de resto	x %= y	x = x % y
Atribuição exponencial	x **= y	x = x ** y

Operadores de Comparação

- Igualdade (==)
  - Se os valores forem iguais retorna verdadeiro
    - Primero converte o operando se não for do mesmo tipo, então aplica a comparação estrita.

```
1   1    ==  1          // verdade
2   '1'  ==  1          // verdade
3   1    == '1'         // verdade
4   0    == false       // verdade
5   0    == null        // falso
6   var object1 = {'key': 'value'}, object2 = {'key': 'value'};
7   object1 == object2 // falso
8   0    == undefined   // falso
9   null == undefined   // verdade
```

Objetos JavaScript não podem ser comparados. A comparação de dois objetos JavaScript sempre retorna false.

Operadores de Comparação

- Desigualdade (!=)
  - Retorna verdadeiro se os valores dos operandos são diferentes
    - Se os operando não forem do mesmo tipo, primeiro converte os tipos e depois realiza a comparação.

```
1    1 !=   2      // verdade
2    1 !=  '1'     // falso
3    1 !=  "1"     // falso
4    1 !=  true    // falso
5    0 !=  false   // falso
```

Operadores de Comparação

- Identidade/igualdade estrita (===)
  - Retorna verdadeiro se os valores e tipos dos operandos são iguais

```
1   3 === 3   // verdade
2   3 === '3' // falso
3   var object1 = {'key': 'value'}, object2 = {'key': 'value'};
4   object1 === object2 //falso
```

Objetos JavaScript não podem ser comparados. A comparação de dois objetos JavaScript sempre retorna false.

Operadores de Comparação

- Non-identity/desigualdade estrita (!==)
  - Retorna verdadeiro se os valores e tipos dos operandos não são iguais

```
1    3 !== '3' // verdade
2    4 !== 3   // verdade
```

Operadores Relacionais

Operador	Sintaxe
Maior que (>)	4 > 3 //verdade
Maior ou igual (>=)	5 >= 1 //verdade 3 >= 3 //verdade
Menor que (<)	2 < 5 //verdade
Menor ou igual (<=)	2 <= 3 //verdade 2 <= 2 //verdade

Operadores Lógicos

Operador	Utilização	Descrição
AND (&&)	expr1 && expr2	Retorna verdadeiro caso as duas expressões sejam verdadeiras.
OR (||)	expr1 || expr2	Retorna falso se as duas expressões forem falsas.
NOT (!)	!expr	Negação lógica. Retorna falso caso a expressão possa ser convertida para verdadeiro, caso contrário, retorna verdadeiro.

Operadores Lógicos

- Exemplos do operador lógico AND

```
1    var a1 =  true && true;      // t && t retorna true
2    var a2 =  true && false;     // t && f retorna false
3    var a3 = false && true;      // f && t retorna false
4    var a4 = false && (3 == 4); // f && f retorna false
```

Operadores Lógicos

- Exemplos do operador lógico OR

```
1    var o1 =  true || true;     // t || t retorna true
2    var o2 = false || true;     // f || t retorna true
3    var o3 =  true || false;    // t || f retorna true
4    var o4 = false || (3 == 4); // f || f retorna false
```

Operadores Lógicos

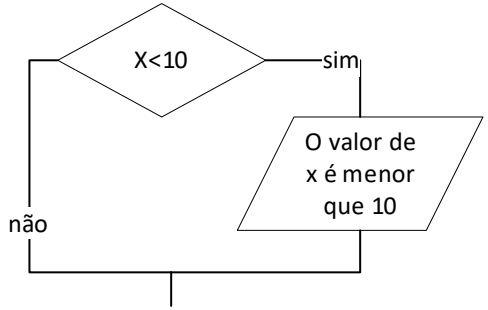
- Exemplos do operador lógico NOT

```
1    var n1 = !true;   // !t retorna false
2    var n2 = !false;  // !f retorna true
```

Estruturas de Controle de Fluxo

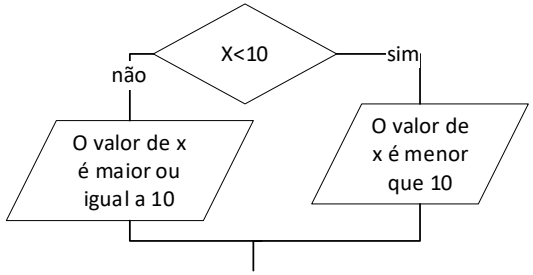
- IF

```
if( x < 10 ){
    alert("O valor de x é menor que 10");
}
```



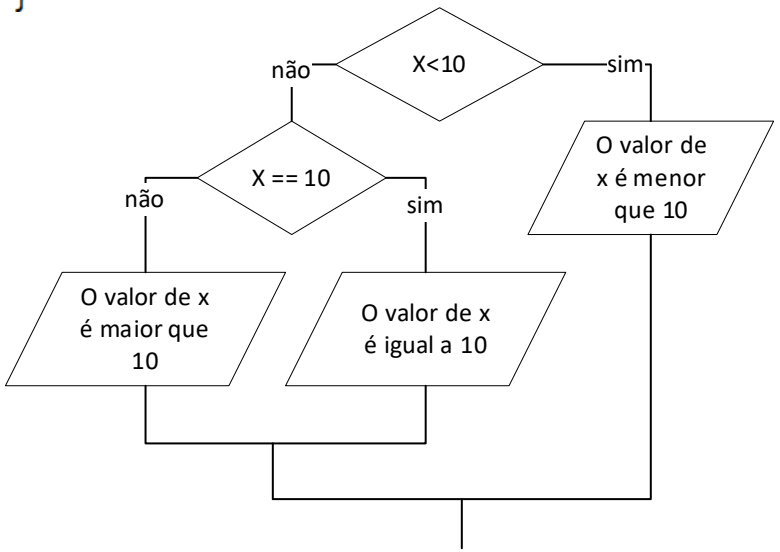
- IF ELSE

```
if( x < 10 ){
    alert("O valor de x é menor que 10");
}else{
    alert("O valor de x é maior ou igual a 10");
}
```



- ELSE IF

```
if( x < 10 ){
    alert("O valor de x é menor que 10");
}else if(x == 10){
    alert("O valor de x é igual a 10");
}else{
    alert("O valor de x é maior que 10");
}
```



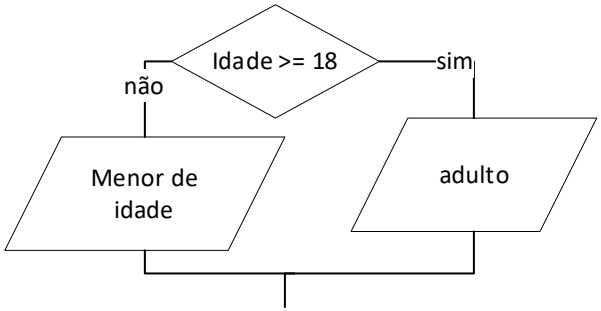
Operador condicional ternário

- Equivale ao IF Else

- Sintaxe:

<expressão> ? <true> : <false>

- Exemplo:
  - var status = (idade >= 18) ? "adulto" : "menor de idade";

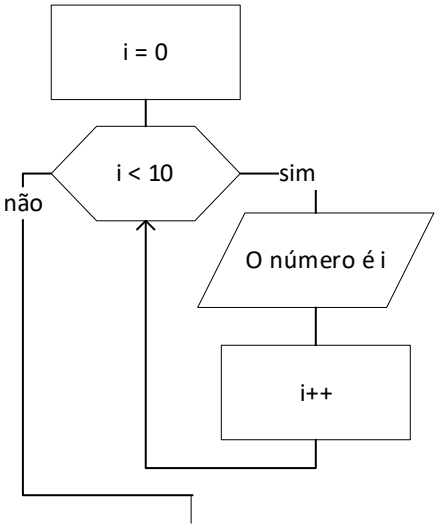


Estruturas de Repetição

- For

```
1   var i;
2
3   for(i = 0; i < 5; i++){
4       console.log("O número é: " + i);
5   }
```

```
for (i = 0; i < 5; i++) {
    text += "O número é " + i + "<br>";
}
document.getElementById("minha-div").innerHTML = text;
```



- While

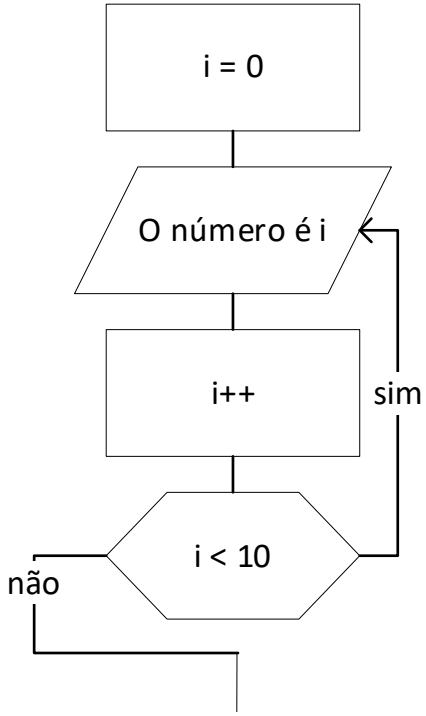
```
1   var i = 0;
2
3   while(i < 10){
4       console.log("O número é: " + i);
5       i++;
6   }
```

```
while (i < 10) {
    text += "<br>O número é " + i;
    i++;
}
document.getElementById("minha-div").innerHTML = text;
```

Estruturas de Repetição

- Do/While

```
1    var i = 0;
2
3    do{
4        console.log("O número é: " + i);
5        i++;
6    }while(i < 10);
```




```
do {
    text += "<br>O número é " + i;
    i++;
} while(i < 10);
document.getElementById("minha-div").innerHTML = text;
```

Estruturas de Controle de Fluxo

- **Switch**

```
switch (new Date().getDay()) {
    case 0:
        day = "Domingo";
        break;
    case 1:
        day = "Segunda";
        break;
    case 2:
        day = "Terça";
        break;
    case 3:
        day = "Quarta";
        break;
    case 4:
        day = "Quinta";
        break;
    case 5:
        day = "Sexta";
        break;
    case 6:
        day = "Sábado";
}
```

```
console.log(day);
```

Funções

- Assim como em outras linguagens de programação, como o C, uma função é um bloco de código designado para uma determinada tarefa.

```
function name(){
    //código a ser executado
}
```

- Uma função é executada quando ela é chamada ("invocada")
  - Exemplo:

```
name();
```

Funções

- As funções podem receber uma lista de parâmetros que são informados entre os parênteses:

```
function name(parametro1, parametro2, parametro3){
    //código a ser executado
}
```

Lembre-se que a linguagem JavaScript é fracamente tipada, isto significa que na lista de parâmetros também não é informado o tipo das variáveis.

- Quando o JavaScript atinge uma instrução return, a função para de ser executada. O valor de retorno é retornado de volta ao chamador

return parâmetro 1 + parâmetro 2 + parametro3