

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?

```
<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>
```

Dentro da tag HEAD

Inseri o título

Inseri o parágrafo 1
Inseri o parágrafo 2

No corpo da página

No arquivo externo

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

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

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>
```

Inseri o título

Inseri o parágrafo 1

Inseri o parágrafo 2

```
// 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");
```

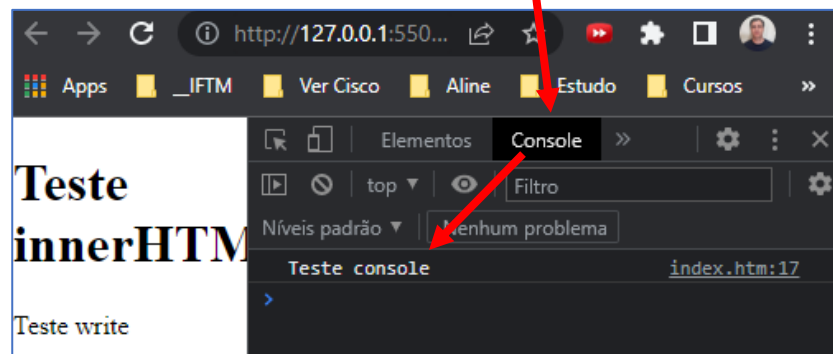
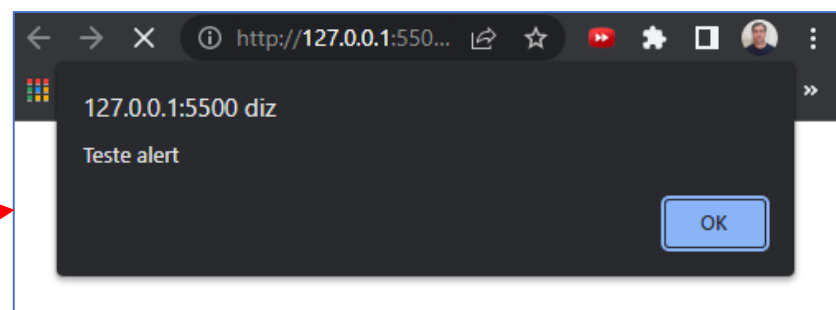
```
    console.log("Teste console");
```

```
</script>
```

Tecla F12

Teste innerHTML

Teste write



Comentários

- Pode ser feito de linha ou de bloco.

```
1  //este é um comentário de linha
2
3  /*
4     este é um
5     comentário
6     de
7     bloco
8  */
```

Variáveis

- São containeres 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);
```

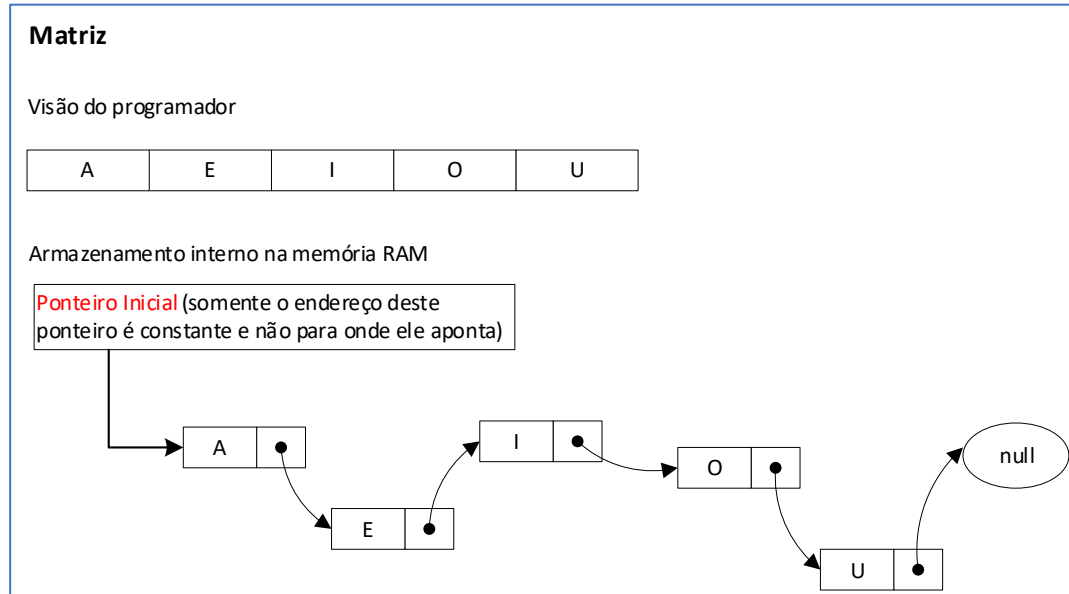


Variáveis – outras características

- 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

- Primeiro 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

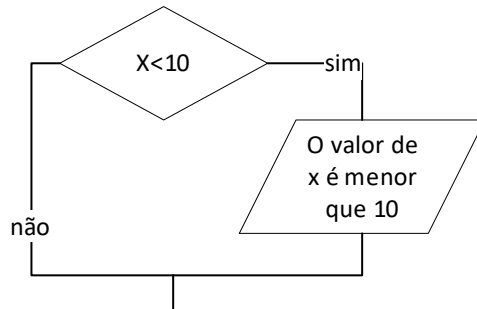
- Exemplos do operador lógico NOT

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

Estruturas de Controle de Fluxo

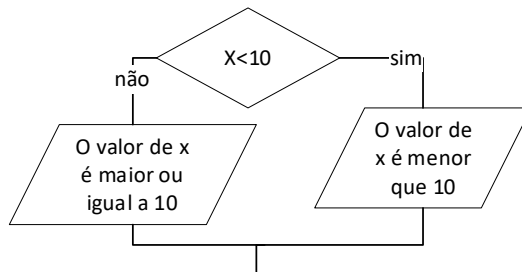
• IF

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



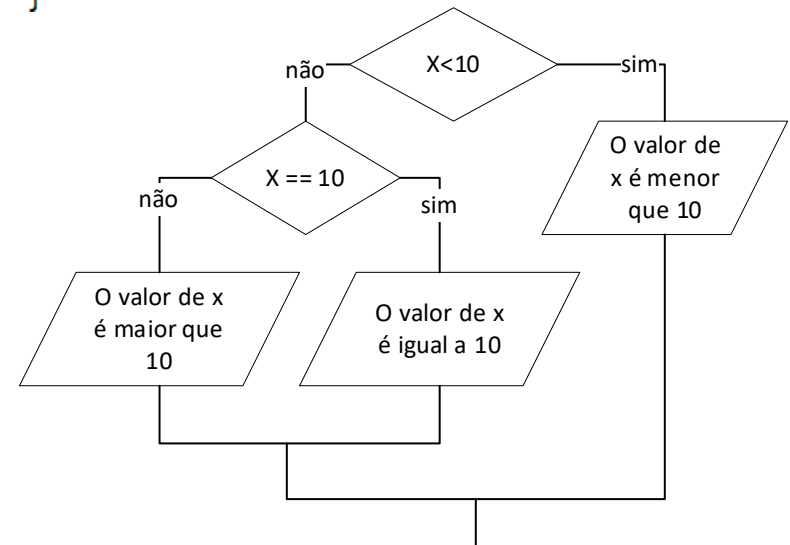
• IF ELSE

```
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

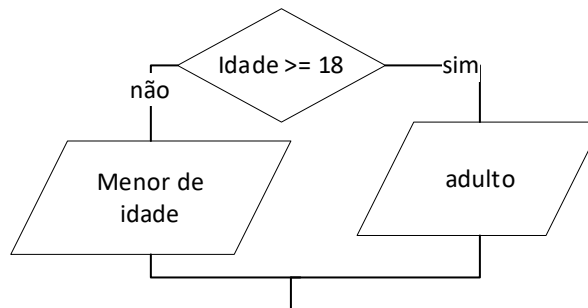
- Equivale ao IF Else

- Sintaxe:

`<expressão> ? <true> : <false>`

- Exemplo:

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

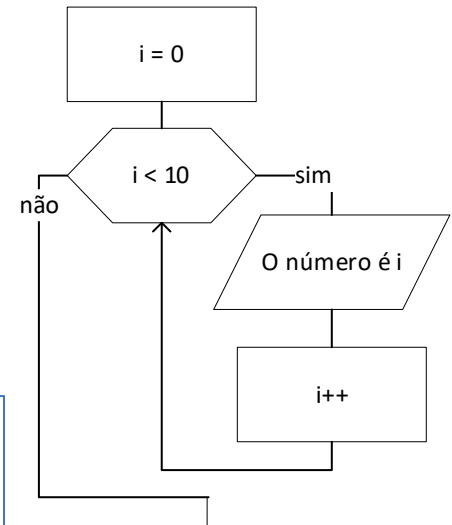


Estruturas de Repetição

• 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  }
```

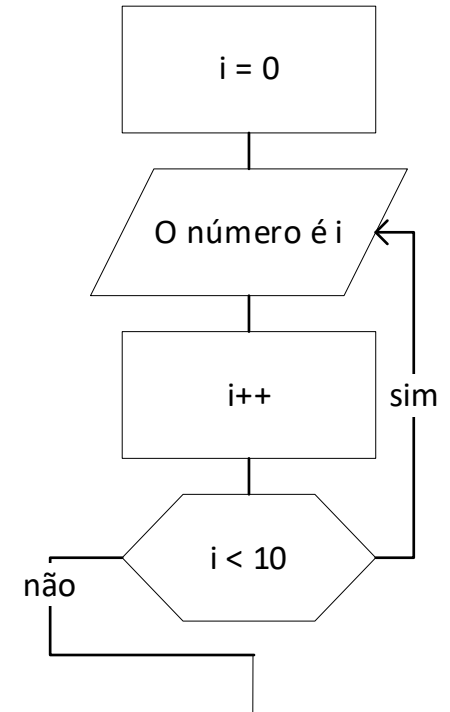
```
while (i < 10) {  
    text += "<br>O número é " + i;  
    i++;  
}  
document.getElementById("minha-div").innerHTML = text;
```

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

console.log(day);

```
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";  
}
```

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

Operadores de tipo JavaScript

Operador	Descrição
typeof	Retorna o tipo de variável
instanceof	Retorna verdadeiro se um objeto for uma instância de um tipo de objeto

```
var meuTeste = function () { };  
var nome = "Maria";  
var tamanho = 162;  
const hoje = new Date();
```

```
alert(typeof meuTeste);    // retorna function  
alert(typeof nome);       // retorna string  
alert(typeof tamanho);    // retorna number  
alert(typeof hoje);       // retorna object  
alert(typeof naoTem);     // retorna undefined  
alert(hoje instanceof Date); // retorna True
```

Tipos de dados

- JavaScript tem tipos dinâmicos. Isso significa que a mesma variável pode ser usada para armazenar diferentes tipos de dados.
- Alguns tipos de dados:
 - Booleanos : `true` e `false`.
 - Matrizes: `const carros = ["Fusca", "Passat", "Belina"];`
 - Undefined: uma variável sem valor tem o valor `undefined`. O tipo também é `undefined`.
 - Números : `42` ou `3.14159`.
 - String: `"Texto"`. Pode usar aspas simples ou duplas
 - Objetos : `const pessoa = {nome:"João", idade:50};`

Tipos de dados - Matrizes

- Variáveis JavaScript podem ser objetos.
- Arrays (matrizes) são tipos especiais de objetos.
- Pode ter variáveis de tipos diferentes no mesmo Array:
 - objetos
 - funções
 - arrays

```
var myFunction = function () { };  
var hoje = new Date();  
const carros = ["Fusca", "Passat", "Belina"];  
  
const myArray = [];  
myArray[0] = hoje;    //objeto  
myArray[1] = myFunction; //função  
myArray[2] = carros;  //matriz
```

Tipos de dados - Objetos

```
const person = {  
  firstName: "John",  
  lastName: "Doe",  
  id: 5566,  
  fullName: function() {  
    return this.firstName + " " + this.lastName;  
  }  
};
```

- Objetos podem conter muitos valores
- Valores são escritos como pares
nome:valor
- Os pares name:values são chamados de propriedades
- Acessando Propriedades
person.lastName;
ou
person["lastName"];
- Acessando métodos
name = person.fullName();

A comparação de dois objetos JavaScript sempre retorna false.

Quando usar matrizes ou objetos?

Muitas linguagens de programação suportam arrays com índices nomeados. Arrays com índices nomeados são chamados de arrays associativos (ou hashes). JavaScript não suporta arrays associativos.

- Você deve usar objetos quando quiser que os nomes dos elementos sejam strings (texto).
- Você deve usar arrays quando quiser que os nomes dos elementos sejam números.

Relembrando como definir matriz e objeto.

Matrizes: `const carros = ["Fusca", "Passat", "Belina"];`

Objetos : `const pessoa = {nome:"João", idade:50};`

Funções de seta

- Permitem escrever uma sintaxe de função mais curta

- Antes da seta

```
hello = function() {  
  return "Hello World!";  
}
```

- Com função de seta

```
hello = () => {  
  return "Hello World!";  
}
```

- Uma instrução e retorna um valor, pode remover as chaves e a palavra-chave return.

```
hello = () => "Hello World!";
```

- Se tiver parâmetros
hello = (val) => "Hello " + val;

- Apenas um parâmetro retire os parênteses
hello = val => "Hello " + val;

Vetores

```
var carros = ["BMW", "Volvo", "Saab", "Ford"];  
var i = 0;  
var text = "";  
  
while (carros[i]) {  
    text += carros[i] + "<br>";  
    i++;  
}  
document.getElementById("minha-div").innerHTML = text;
```

Percorrendo com
WHILE

Percorrendo com
FOR

```
var carros = ["BMW", "Volvo", "Saab", "Ford"];  
var i = 0;  
var text = "";  
  
for(i = 0; i < carros.length; i++){  
    text += carros[i] + "<br>";  
}  
document.getElementById("minha-div").innerHTML = text;
```

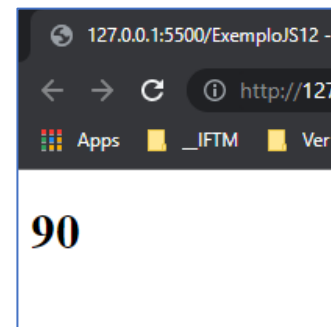
forEach()

- Em matriz, chama uma função uma vez para cada elemento da matriz

```
<script>
  window.onload = funcao01;
  function funcao01() {
    const numeros = [10, 30, 50];
    let soma = 0;

    numeros.forEach(funcao);

    function funcao(value) {
      soma += value;
    }
    document.getElementById("id01").innerHTML = soma;
  }
</script>
<h1 id="id01"></h1>
```

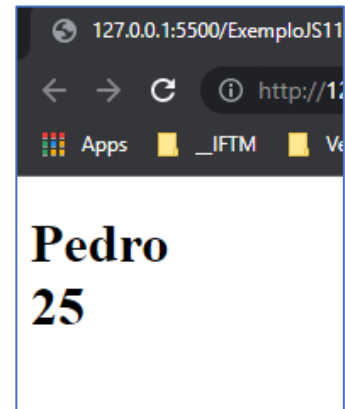


For In

- Em objeto, percorre as propriedades

```
<script>
  window.onload = funcao01;
  function funcao01() {
    const pessoa = {
      nome: "Pedro",
      idade: 25
    };
    let text = "";

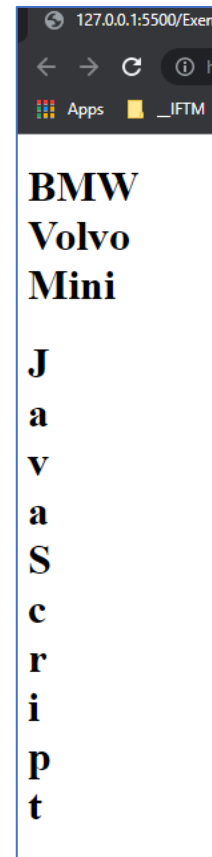
    for (let x in pessoa) {
      text += pessoa[x] + "<br>";
    }
    document.getElementById("id01").innerHTML = text;
  }
</script>
<h1 id="id01"></h1>
```



for (variável of iteravel)

- Permite fazer um loop sobre estruturas de dados iteráveis

```
window.onload = funcao01;  
function funcao01() {  
  const cars = ["BMW", "Volvo", "Mini"];  
  let text = "";  
  for (let x of cars) {  
    text += x + "<br>";  
  }  
  document.getElementById("id01").innerHTML = text;  
  
  let language = "JavaScript";  
  text = "";  
  for (let x of language) {  
    text += x + "<br>";  
  }  
  document.getElementById("id02").innerHTML = text;  
}  
</script>  
<h1 id="id01"></h1>  
<h1 id="id02"></h1>
```

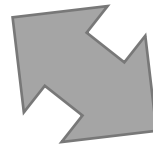


Objeto String

- JavaScript trata valores primitivos como objetos ao executar métodos e propriedades.
- Retorna a quantidade de caracteres de uma string

- **length**

```
var txt = "ABCD"  
var tam = txt.length
```



A variável "tam" receberá o valor 4.

Objeto String

- **slice** ou **substring()**: Extrai uma parte de uma string e retorna a parte extraída em uma nova string.
 - Recebe dois parâmetros: a posição inicial e a posição final (não inclusa)

```
var txt = "Melancia, Abacate, Ameixa"  
var res = txt.slice(4, 8);
```

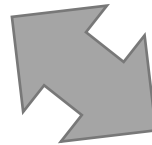


A variável "res" receberá o valor "ncia".

Objeto String

- **substr()**: Extrai uma parte de uma string e retorna a parte extraída em uma nova string.
 - Recebe dois parâmetros: a posição inicial e o segundo parâmetro especifica o comprimento da parte extraída.

```
var txt = "Melancia, Abacate, Ameixa"  
var res = txt.substr(4, 8);
```



A variável "res" receberá o valor "ncia, Ab".

Objeto String

- **replace():** Recebe uma string e onde houver a sua ocorrência, substitui pelo termo informado no segundo parâmetro.
 - Recebe dois parâmetros: a string a ser encontrada e a string que representa o novo valor.
 - substitui apenas a primeira correspondência.

```
var txt = "Visite o CEFET em Uberaba"  
var res = txt.replace("CEFET", "IFTM");
```

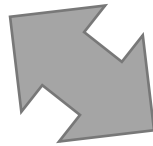


A variável "res" receberá o valor "Visite o IFTM em Uberaba".

Objeto String

- **concat()**: Junta duas ou mais strings.

```
var txt1 = "Olá";  
var txt2 = "Mundo";  
var res = txt1.concat(" ", txt2);
```

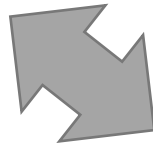


A variável "res" receberá o valor "Olá Mundo".

Objeto String

- **trim()**: Remove os espaços em branco que envolvem os lados da string.

```
var txt = "  Olá Pessoa!!!  ";  
var res = txt.trim();
```



A variável “res” receberá o valor “Olá Pessoa!!!”, isto é, sem os espaços.

Objeto String

- **split**: divide uma string em pedaços armazenando-os em um vetor de string (array de string).

```
var txt = "Como vai você?";  
var res = txt.split(" ");
```

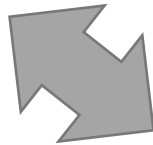


A variável "res" receberá cada parte da string, sendo o resultado:
res[0] == "Como"; res[1] == "vai"; res[2] == "você?";

Objeto String

- O método `indexOf()` retorna o índice (posição) da primeira ocorrência do termo especificado.

```
var txt = "Vamos pensar mais, mais alegremente!"  
var pos = txt.indexOf("mais");
```

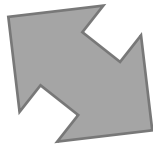


A variável "pos" receberá o valor 13.

Objeto String

- O método **lastIndexOf()** retorna o índice (posição) da última ocorrência do termo especificado.

```
var txt = "Vamos pensar mais, mais alegremente!"  
var pos = txt.lastIndexOf("mais");
```

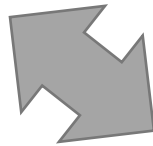


A variável "pos" receberá o valor 19.

Objeto String

- **startsWith**: determina se uma string começa ou não com um determinado termo passado como parâmetro.
 - É importante notar que este método é case-sensitive, isto é, diferencia maiúsculas e minúsculas.

```
var txt = "Como vai você?";  
var res = txt.startsWith("Como");
```

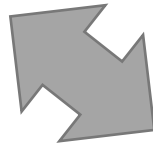


A variável "res" receberá o valor TRUE. Todavia, se não iniciar, receberá o valor FALSE

Objeto String

- **endsWith**: determina se uma string finaliza ou não com um determinado termo passado como parâmetro.
 - É importante notar que este método é case-sensitive, isto é, diferencia maiúsculas e minúsculas.

```
var txt = "Como vai você?";  
var res = txt.endsWith("?");
```



A variável "res" receberá o valor TRUE. Todavia, se não finalizar, receberá o valor FALSE

Objeto String

- **toUpperCase():** converte todos os caracteres de uma string em maiúsculo.
- **toLowerCase():** converte todos os caracteres de uma string em minúsculo.
- **charAt(posição)**
 - Retorna o caractere de uma posição específica da string informada como parâmetro.

Objeto String

- Você pode conferir esses e mais métodos em:

https://www.w3schools.com/jsref/jsref_obj_string.asp

Métodos úteis

- `prompt(<digite seu texto aqui>):`
 - Requer um valor do usuário;
 - Para utilizar o valor deve-se atribuí-lo a uma variável;
 - Para realizar operações matemáticas é necessário converter o valor.
- `parseInt(<valor>):` converte uma string em um inteiro.
- `parseFloat(<valor>):` converte uma string em um número de ponto flutuante.
- `String():` converte o valor em uma string.

Objeto Data

- Algumas formas de criar objeto data:

```
const d = new Date(); //data e hora atuais
```

```
const d = new Date(2018, 11, 24, 10, 33);  
// ano, mês, dia, hora e minuto
```

```
const d = new Date(2018, 11, 24);  
// ano, mês e dia
```

Métodos de Data

- `toString()`:

Mon Oct 10 2022 08:49:04 GMT-0300 (Horário Padrão de Brasília)

- `toUTCString()`:

Mon, 10 Oct 2022 11:49:46 GMT

- `toDateString()`:

Mon Oct 10 2022

Métodos de Data

Método	Descrição
now()	Obter a hora atual. (O número de milissegundos desde a meia-noite de 1º de janeiro de 1970 00:00:00 UTC.)
getFullYear()	Obtenha o ano como um número de quatro dígitos (yyyy)
getMonth()	Obtenha o mês como um número (0-11)
getDate()	Obtenha o dia como um número (1-31)
getDay()	Obtenha o dia da semana como um número (0-6)
getHours()	Obtenha a hora (0-23)
getMinutes()	Obter o minuto (0-59)
getSeconds()	Obter o segundo (0-59)
getMilliseconds()	Obter o milissegundo (0-999)
getTime()	Obter o tempo (milissegundos desde 1 de janeiro de 1970)

Quando se compara datas o resultado é em milissegundos.
Para converter em dias retire os horários dividindo por
(1000 milissegundos * 60 segundos * 60 minutos * 24 horas)

Métodos de Data

Método	Descrição:
setDate()	Definir o dia como um número (1-31). Também pode ser usado para adicionar dias a uma data
setFullYear()	Definir o ano (opcionalmente mês e dia)
setHours()	Definir a hora (0-23)
setMilliseconds()	Definir os milissegundos (0-999)
setMinutes()	Definir os minutos (0-59)
setMonth()	Definir o mês (0-11)
setSeconds()	Definir os segundos (0-59)
setTime()	Definir a hora (milissegundos desde 1 º de janeiro de 1970)

- Um ótimo guia de referência para trabalhar com objetos de Data esta contido em:
https://www.w3schools.com/jsref/jsref_obj_date.asp

Métodos Matemáticos

Método	Descrição
Math.round(x)	Retorna x arredondado para o inteiro mais próximo
Math.trunc(x)	Retorna a parte inteira de x
Math.pow(x, y)	retorna o valor de x elevado a y
Math.sqrt(x)	retorna a raiz quadrada de x
Math.abs(x)	retorna o valor absoluto (positivo) de x
Math.min() e Math.max()	encontra o valor mais baixo ou mais alto em uma lista de argumentos
Math.random()	retorna um número aleatório entre 0 (inclusive) e 1 (exclusivo)

- A linguagem JavaScript oferece uma variedade de métodos pré-definidos para se trabalhar com tarefas matemáticas, confira:

https://www.w3schools.com/jsref/jsref_obj_math.asp

Métodos de Matrizes

No início

shift(): remove o primeiro elemento e retorna o valor que foi extraído.

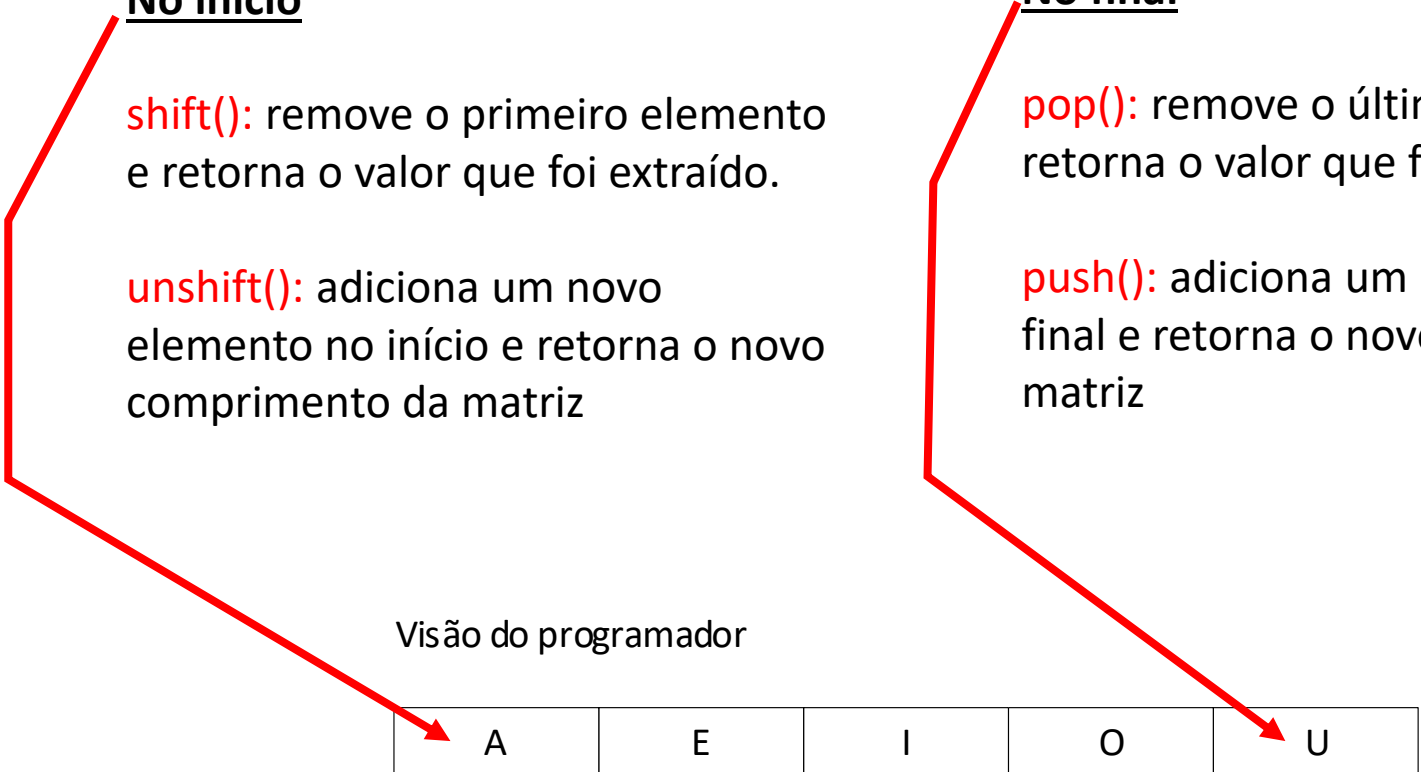
unshift(): adiciona um novo elemento no início e retorna o novo comprimento da matriz

No final

pop(): remove o último elemento e retorna o valor que foi extraído

push(): adiciona um novo elemento no final e retorna o novo comprimento da matriz

Visão do programador



A	E	I	O	U
---	---	---	---	---

splice(): adicionar ou remover em qualquer posição.