

Programador Web

Algoritmos e Lógica de Programação

Adriel Sales





O que é lógica?

“Parte da filosofia que trata das formas do pensamento em geral (dedução, indução, hipótese, inferência, etc.) e das operações intelectuais que visam à determinação do que é **verdadeiro** ou **falso**.”



Lógica



Tudo
começa
com um
PROBLEMA

Houston, we have a problem.
-Apollo 13 1995



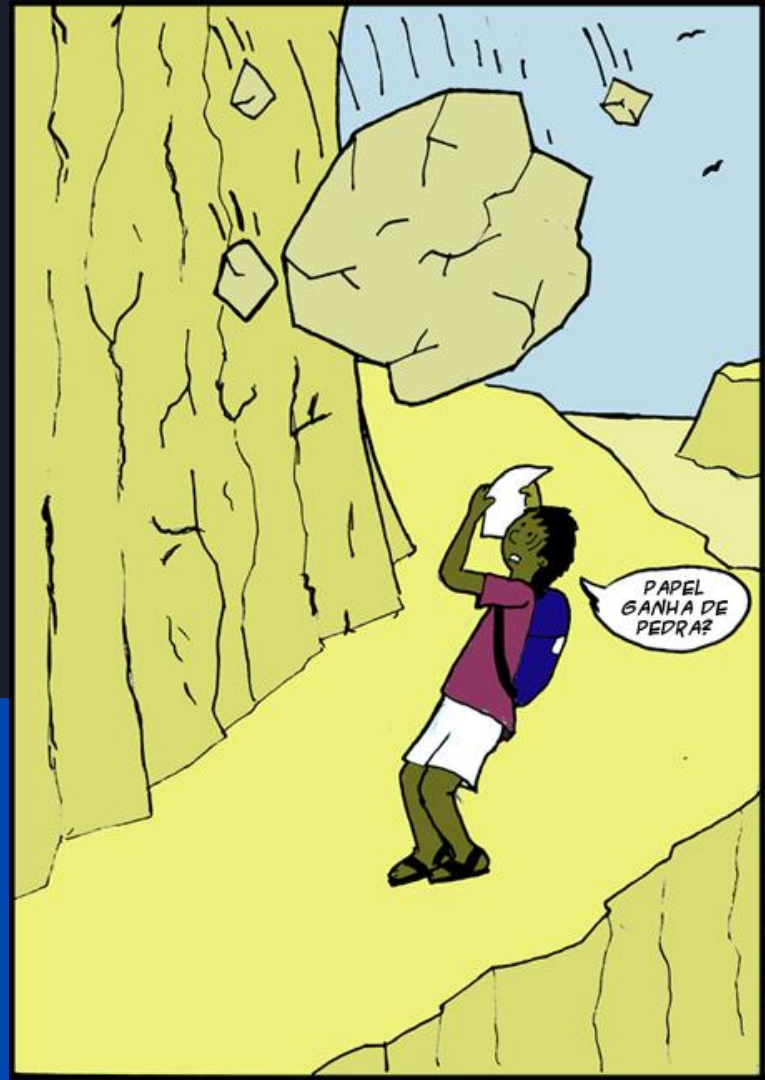
- Questão matemática proposta para ser resolvida.
- Questão difícil, delicada, suscetível de diversas soluções.
- Qualquer coisa de difícil explicação, mistério, enigma.
- Dúvida, questão.

PROBLEMA

Exemplos de problemas

- Trocar uma lâmpada
- Trocar o pneu de um carro
- Preparar o jantar

Porém, sempre que nos deparamos com um **problema**, buscamos sempre uma **SOLUÇÃO!**

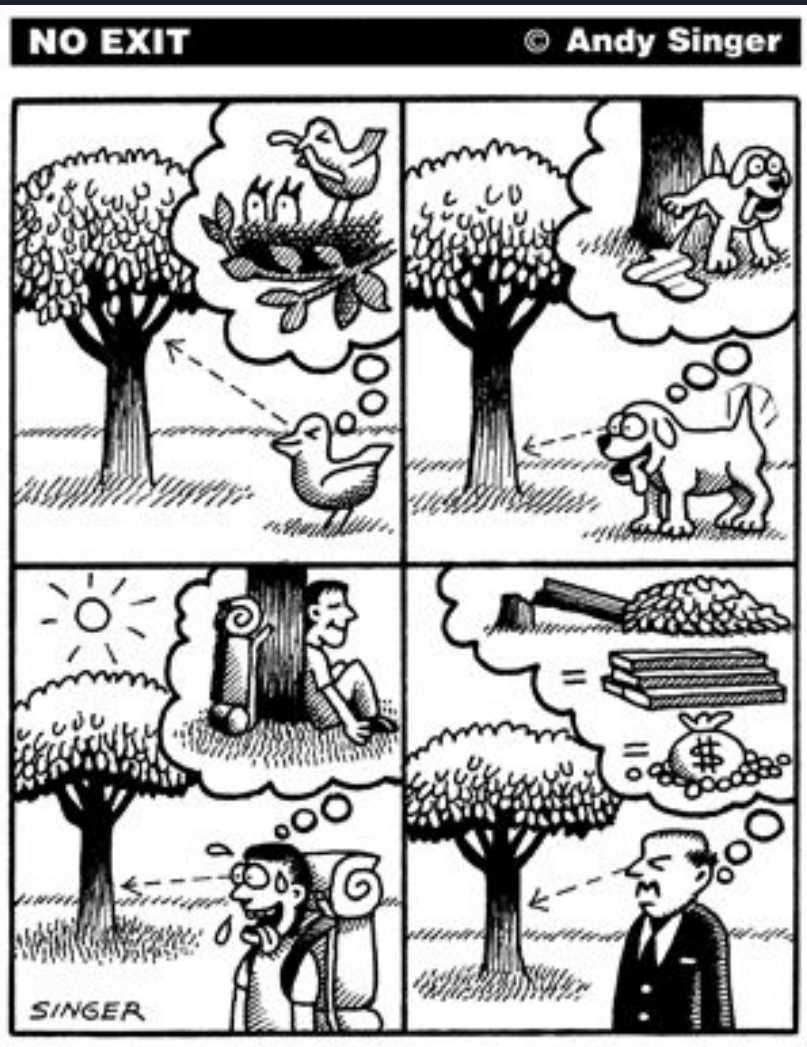


Lembre:

Um problema pode ter várias soluções!



Cada um
pensa de uma
maneira



Lógica de Programação



- **Lógica de Programação**
 - É a técnica de encadear pensamentos para atingir determinado objetivo.
- **Sequência Lógica**
 - São passos executados até atingir um objetivo ou solução de um problema.
- **Instruções**
 - Um conjunto de regras ou normas definidas para a realização de algo.

Ao utilizar lógica \Rightarrow Algoritmos



- **Algoritmo**

- É uma sequência finita de passos que levam a execução de uma tarefa ou solucionar um problema.

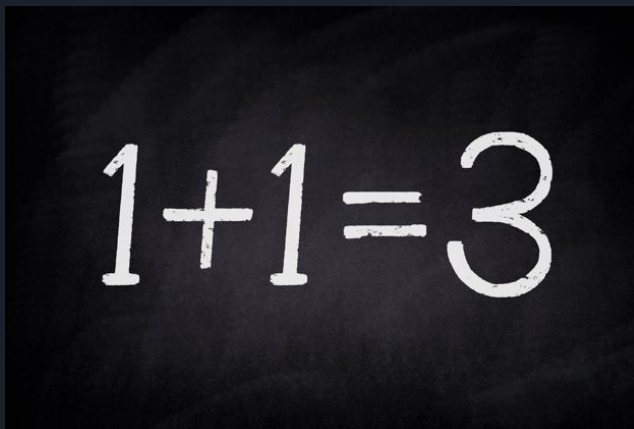
- **Programas**

- São algoritmos escritos em uma linguagem de programação e que são interpretados e executados por uma máquina.

Atenção!

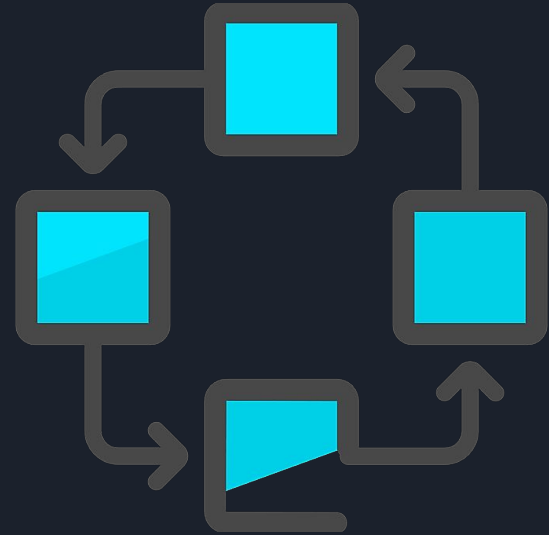
Cuidados devem ser tomados no desenvolvimento de um algoritmo:

Algoritmo Errado \Rightarrow Resultado Errado!


$$1+1=3$$

Métodos de Representação de Algoritmos

- Descrição Narrativa
- Pseudocódigo
- Fluxograma





Descrição Narrativa

Vantagem: simples de ser implementada.

Exemplo:

1. Adquira uma lâmpada nova.
2. Localize a lâmpada a ser trocada.
3. Em seguida, retire a lâmpada queimada e coloque a lâmpada nova.
4. Após a troca, descarte a lâmpada queimada.

Descrição Narrativa

Desvantagem:

pode haver erros ou equívocos na interpretação.





Pseudocódigo

- Utiliza-se de uma linguagem intermediária entre **linguagem falada** e a **linguagem de programação**.
- Deve ser **independente de linguagem de programação** a ser posteriormente utilizada na codificação.

Exemplo de um Pseudocódigo

Algoritmo: Procurar uma foto em um Livro

1. Pegar o Livro;
2. Abrir o Livro;
3. Visualizar a Página;
4. **Se** a foto procurada estiver na página
 1. Anotar o número da página;
 2. Guardar o livro;
 3. Fim.
5. **Se não**
 1. Virar a página;
 2. Voltar ao passo 3;



Laço de Repetição

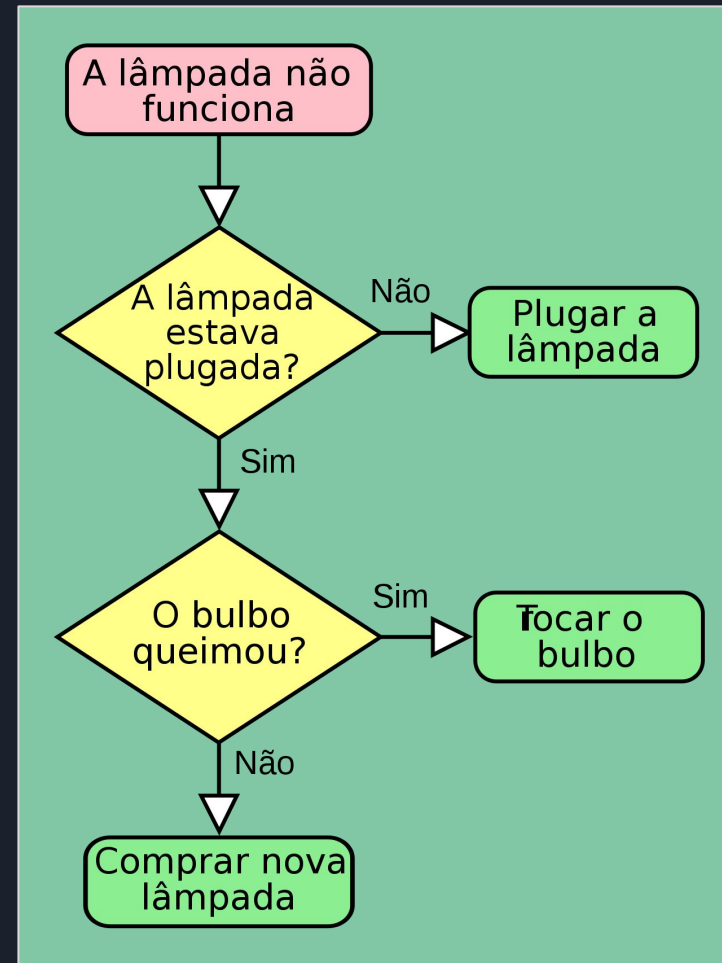
Instruções utilizadas:

Pegar, Abrir, Visualizar, Anotar, Guardar, Virar, Voltar.

ESTRUTURA CONDICIONAL (Se Verdadeiro, Senão)

Fluxograma

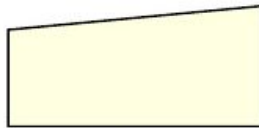
→ Modelo que utiliza figuras para representar o fluxo dos dados e os comandos do algoritmo. Cada operação a ser executada é representada por um símbolo cuja forma identifica o tipo de processo envolvido.



Fluxograma - símbolos básicos



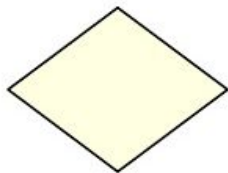
Início ou término
de um fluxograma



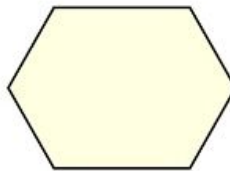
Entrada de dados
via teclado



Procedimento interno
e/ou
Mudança de conteúdo



Tomada de decisão

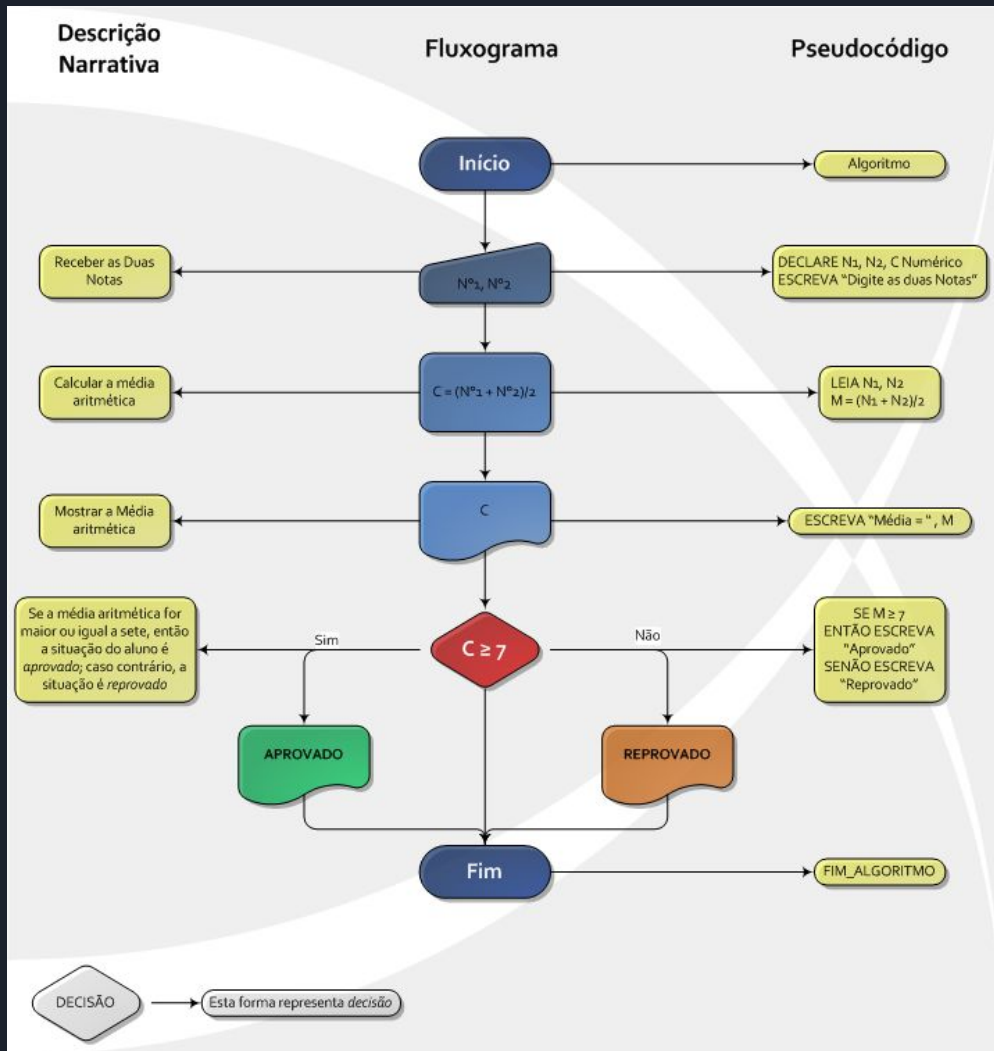


Repetição



Saída de dados
para monitor

Métodos de Representação de Algoritmos

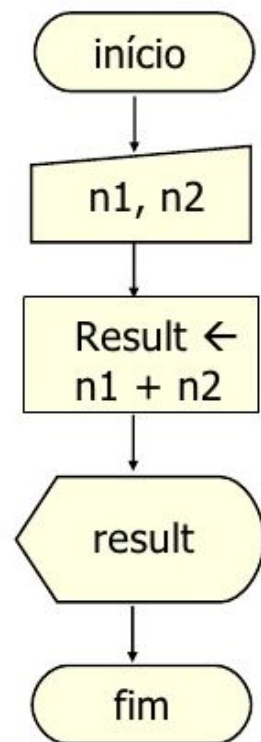


- x **Elaborar o algoritmo que faça a soma de dois números inteiros quaisquer:**

Pseudocódigo

```
algoritmo somadoisnum  
  var n1, n2, result: inteiro  
  
início  
  leia n1  
  leia n2  
  result  $\leftarrow$  n1 + n2  
  escreva result  
  
fim
```

Fluxograma



Linguagem de Programação

```
program somadoisnum;  
  var n1, n2, result: integer;  
  
begin  
  readln(n1);  
  readln(n2);  
  result := n1 + n2;  
  writeln (result);  
  
end.
```

**Primeiro Problema:
algoritmo para trocar
uma lâmpada.**



Primeiro algoritmo: trocando uma lâmpada.

1. Do que precisamos?
2. Como iremos proceder?
3. Qual foi o resultado final?

O que orienta a obtenção dos
procedimentos para a solução é a

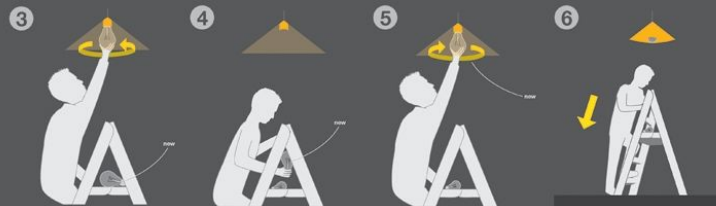
Lógica.

How to change a light bulb



First, please make sure the switch is off, the platform of the ladder is locked, and both hands are dry.

Place the new bulb on the platform of the ladder, climb up the ladder.



Rotate the old bulb in clockwise direction, take it off, and place it on the platform of the ladder.

Take the new bulb

Fix the new bulb by rotating it to the socket in anti-clockwise direction.

Climb down the ladder.



Switch on the light, it should light up if working properly.



Dispose the old light bulb warped by paper.

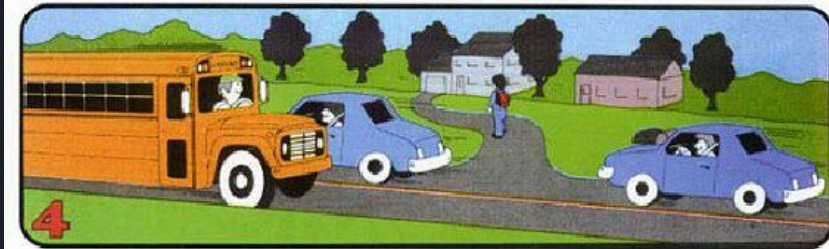
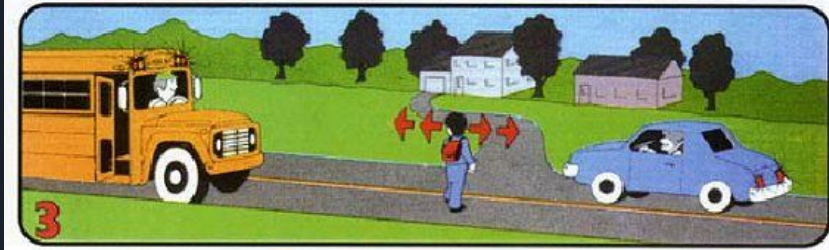
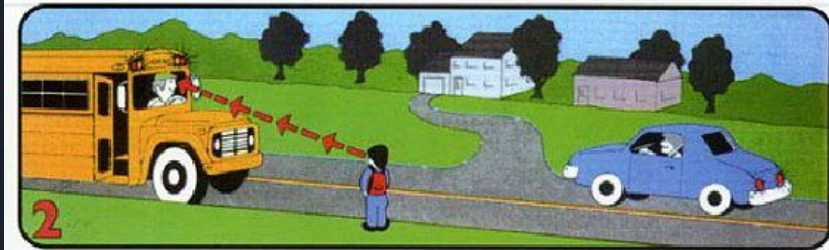
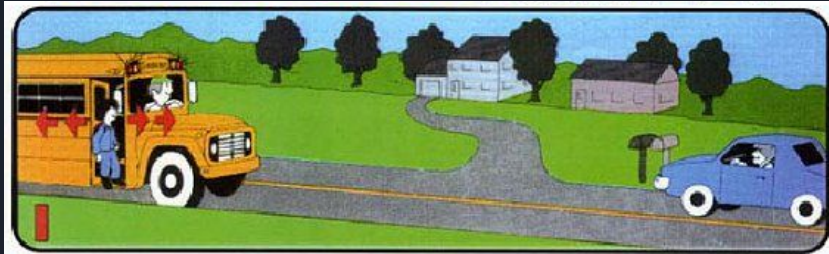
or



Dispose the old light bulb to light bulb recycling bin.

Segundo Problema: algoritmo para cruzar a rua.

1. Do que precisamos?
2. Como iremos proceder?
3. Qual foi o resultado final?





Algoritmos: Conceitos Básicos

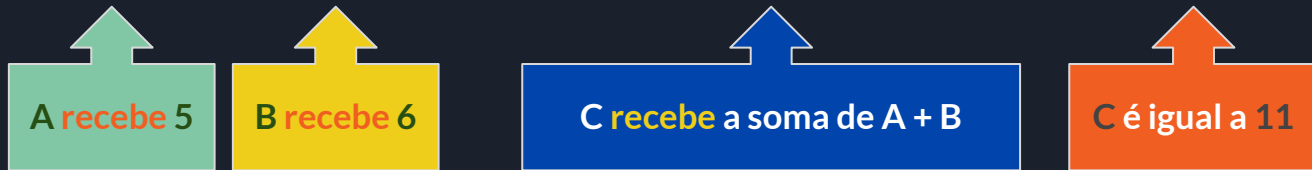
É preciso ter compreensão de alguns conceitos como:

- Constante;
- Variável;
- Identificador;
- Palavra Reservada;
- Entrada;
- Saída.

Algoritmos: Conceitos Básicos

- **Constante:** São endereços de memória destinados a armazenar informações fixas, inalteráveis durante a execução do programa. **Exemplo:** $\pi = 3.1416$.
- **Variável:** São endereços de memória destinados a armazenar informações temporariamente.

◆ Se $A = 5$ e $B = 6$, então, se $C = A + B$, logo $C == 11$.





Algoritmos: Conceitos Básicos

- **Identificadores:** São os nomes que damos as variáveis, constantes e programas: **Exemplo: `var idade`.**
- **Palavras Reservadas:** São identificadores predefinidos que possuem significados especiais para o interpretador do algoritmo.
 - ◆ **`var, if, else, function, round`, entre outras.**

Neste caso, estamos tratando da linguagem JavaScript

Algoritmos: Conceitos Básicos

- **Entrada:** São os dados informados no início ou durante a execução do programa, vindos ou não de um teclado.
- **Saída:** São os resultados do processo da computação, impressos ou exibidos em tela.





Operadores de Atribuição

→ Nome, operador e significado:

◆ Atribuição: $X = Y$ (x recebe y)



Operadores Aritméticos

→ Operação e Símbolo:

◆ Adição +

◆ Subtração -

◆ Multiplicação *

◆ Divisão /



Operadores Relacionais

→ Operação e Símbolo:

- ◆ Igual a ==
- ◆ Diferente de !=, <> ou # (no JS usaremos !=)
- ◆ Maior que >
- ◆ Menor que <
- ◆ Maior ou igual a >=
- ◆ Menor ou igual a <=



Tipos de dados

- Numérico: inteiro (0, 2, 11, -5) ou real (0.10, -5.25, 30.78).
- Literal (**string**): “Somos programadores web!”, “andar”.
- Lógico (**booleano**): true ou false (verdadeiro ou falso)



Operadores Lógicos

→ Operação e descrição:

- ◆ **AND** lógico (&&): Retorna verdadeiro caso ambos operandos sejam verdadeiros; caso contrário, retorna falso.

```
var a1 = true && true;    // t && t retorna true  
var a2 = true && false;   // t && f retorna false
```

Operadores Lógicos

→ Operação e descrição:

- ◆ **OU** lógico (`||`): se utilizado com valores booleanos, retorna verdadeiro caso ambos os operandos sejam verdadeiro; se ambos forem falsos, retorna falso..

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



Operadores Lógicos

→ Operação e descrição:

- ◆ **NOT** lógico (!): Retorna falso caso o único operando possa ser convertido para verdadeiro; senão, retorna verdadeiro.

```
var n1 = !true;    // !t retorna false  
var n2 = !false;   // !f retorna true  
var n3 = !"Gato";  // !t retorna false
```

Precedência Geral dos Operadores Aritméticos



Quando uma **expressão aritmética** precisa ser **avaliada num algoritmo**, o analisador processa a expressão **dando prioridade para certos operadores**. Essa ordem de prioridade na avaliação dos operadores numa expressão aritmética é chamada de **precedência de operadores**.

Precedência Geral dos Operadores Aritméticos



Ordem	Operação	Símbolo
1ª	Parênteses	()
2ª	Potenciação	**
3ª	Multiplicação, Divisão, Resto e Divisão Inteira	*, /, mod, div
4ª	Adição, Subtração	+, -

$$(5+3)**2 * (5-2) + 8$$

$$8**2 * 3 + 8$$

$$64 * 3 + 8$$

$$192 + 8$$

$$200$$



Mais um exemplo

Escreva um algoritmo que armazene o valor 20 em uma variável A e o valor 5 em uma variável B. em seguida, some as variáveis A com B e exiba o resultado (faça a descrição narrativa e o pseudocódigo).

Solução

Algoritmo: somaDoisNumeros

1. **A** recebe **20**.
2. **B** recebe **5**.
3. **Soma** recebe o valor de A somado ao valor de B.
4. Escreva o valor de **Soma**.

Descrição Narrativa

Algoritmo: somaDoisNumeros

var a, b, soma: **inteiro**

a = 20;

b = 5;

soma = a + b;

resultado = soma;

Escreva **resultado**;

Pseudocódigo

Hora de codar JavaScript!

- É a linguagem do browser
- A linguagem mais popular do mundo
- Feita em 1995 para o Netscape 2.0 sob o nome de Mocha depois LiveScript e por fim JavaScript (ECMA Script).



O trio de sucesso do Front End



JavaScript

JS

```
<script type="text/javascript">
```

Tag HTML para JavaScript

Atribuição de valores

Declaração de variáveis

```
var num1 = 10;
```

```
var num2 = parseInt(prompt("Informe um número"));
```

Expressão aritmética

Concatenação de Strings

```
var soma = num1 + num2;
```

```
document.write("<h1>A soma é " + soma + "</h1>");
```

```
</script>
```

Parâmetro da função `write()`;

Obs.: funções podem receber 0 ou N parâmetros

Instruções acabam com ponto e vírgula (;)

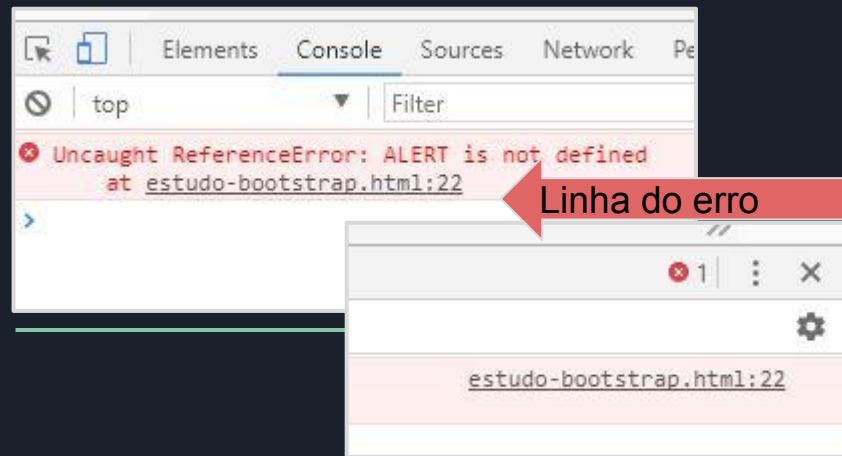
JavaScript é Case Sensitive

JS

`alert("hellow word!");`

FUNCIONA

`ALERT("hellow word!");`



NÃO FUNCIONA

JavaScript: declaração Constantes e Variáveis

JS

```
const HORAS_DO_DIA = 24;
```

```
var semCaracteresEspeciais;
```

```
var sem_Espaços_no_meio;
```

```
var sem_numeros_no_inicio_01;
```

```
const horas_do_dia = 24;
```

```
var @&* &* &"" %;
```

```
var minha variável-x;
```

```
var 123;
```

É aceitável usar \$ ou _ no início do nome. Ex: \$idade, _idade;

CORRETO

ERRADO



Funções com JavaScript

A yellow circle containing the letters 'JS' in a bold, black, sans-serif font.

Uma função JavaScript é um bloco de código projetado para executar uma tarefa específica.

```
function nomeDaFuncao(){  
    //ações da função  
    document.write("<br>");  
}
```

Obs.:
As variáveis declaradas dentro dos parênteses pertencem ao corpo/escopo da função e só existem localmente, ou seja, dentro dela.

`nomeDaFuncao();` /*lembre-se de chamar a função*/

Funções com JavaScript

A yellow circle containing the letters "JS" in a bold, black, sans-serif font, representing JavaScript.

Uma função pode chamar uma outra função.

```
function pularLinha(){  
    document.write("<br>");  
}  
  
function imprime(conteudo){  
    document.write(conteudo);  
    pularLinha();  
}
```

Funções com JavaScript

A yellow circle containing the letters 'JS' in a bold, black, sans-serif font, representing JavaScript.

Uma função pode retornar valores.

```
function minhaFuncaoParaSomar(valor1, valor2){  
    return valor1 + valor2;  
}  
  
var retorno = minhaFuncaoParaSomar(5,9);
```

Strings em JavaScript

A yellow circle containing the letters "JS" in a bold, black, sans-serif font, representing JavaScript.

Strings em
JavaScript **são**
usadas para
armazenar e
manipular texto.

```
/*Tamanho do texto (string)*/
var meuTexto = "Em programação javascript, um texto é uma string.";
/*Tamanho do texto (string)*/
var tamanhoDoTexto = meuTexto.length;
imprime("O texto possui " + tamanhoDoTexto + " caracteres.");
//texto em minuscúlo
imprime(meuTexto.toLowerCase());
//texto em maiúsculo
imprime(meuTexto.toUpperCase());
//usando a função substring para pegar uma parte da string
imprime(meuTexto.substring(13,25));
```

Strings em JavaScript

A yellow circle containing the letters "JS" in a bold, black, sans-serif font, representing JavaScript.

O método **replace ()** substitui um valor especificado com outro valor em uma string:

```
/*substituindo uma parte da string*/  
imprime(meuTexto.replace("javascript", "web"));
```

https://www.w3schools.com/js/js_string_methods.asp

JavaScript: Blocos Condicionais

A yellow circle containing the letters "JS" in a bold, black, sans-serif font, representing JavaScript.

São estruturas que recebem um ou mais valores como parâmetros e, a partir de comparações lógicas, retornam **true** ou **false**. A condicional mais comum é o **if** e possui a seguinte estrutura:

```
if (valorQueEstouPassando == true) {  
    imprime("Verdadeiro");  
} else {  
    imprime("Falso");  
}
```

JavaScript: Blocos Condicionais

A yellow circle containing the letters "JS" in a bold, black, sans-serif font.

Ainda é possível utilizar os operadores **&&** (E) e o **||** (OU):

```
if (idade >= 18 && temCarteira) {  
    alert("Pode dirigir");  
} else {  
    alert("Não pode");  
}
```

JavaScript: Blocos Condicionais aninhados

A yellow circle containing the letters 'JS' in a bold, dark blue font.

Se necessário testar mais de uma condição podemos **aninhar ifs** da seguinte maneira:

```
if (hora >= 0 && hora < 12) {  
    saudacao = "bom dia!";  
} else if (hora >= 12 && hora < 18) {  
    saudacao = "Boa tarde!";  
} else {  
    saudacao = "boa noite!";  
}
```

JavaScript: Blocos Condicionais aninhados

JS

Cuidado com muitos ifs...

Condition



```
if (corDoCarro == "Branco") {  
    imprime("Valor do carro: " + 25.000);  
} else if (corDoCarro == "Vermelho") {  
    imprime("Valor do carro: " + 27.000);  
} else if (corDoCarro == "Cinza") {  
    imprime("Valor do carro: " + 30.000);  
} else if (corDoCarro == "Preto") {  
    imprime("Valor do carro: " + 35.000);  
} else {  
    imprime("Valor do carro: " + 15.000);  
}
```


Prefira utilizar o Switch



```
switch (corDoCarro) {  
  case "Branco":  
    imprime("Valor do carro: " + 25.001);  
    break;  
  case "Vermelho":  
    imprime("Valor do carro: " + 27.001);  
    break;  
  case "Cinza":  
    imprime("Valor do carro: " + 30.001);  
    break;  
  case "Preto":  
    imprime("Valor do carro: " + 35.001);  
    break;  
  default:  
    imprime("Valor do carro: " + 15.001);  
}
```

JavaScript: Loops (laços de repetição)

A yellow circle containing the letters "JS" in a bold, black, sans-serif font, representing JavaScript.

Loops são estruturas que podem executar um bloco de código um número “n” de vezes. Usaremos os seguintes loops em JavaScript:

- **For**
 - Usado quando sabemos a quantidade de vezes do loop.
- **While**
 - Usado quando sabemos ou não a quantidade de vezes do loop.
- **Do while**
 - Usado quando precisamos rodar o loop antes de verificar se a condição é verdadeira.

JavaScript: laço de repetição “for”

JS

```
for (declaração antes do loop; condição; incremento/decremento) {  
    bloco de código a ser executado...  
}
```

```
/*Loop crescente de 1 a 10*/  
for (var i = 1; i <= 10; i++) {  
    imprime("Loop incremental de 1 até " + i);  
}  
  
/*Loop decrescente de 10 a 1*/  
for (var i = 10; i >= 1; i--) {  
    imprime("Loop decremental de 10 até " + i);  
}
```

JavaScript: laço de repetição “while”

JS

```
while (condição) {
```

bloco de código a ser executado...

```
}
```

```
/*Loop crescente de 1 a 10*/
```

```
var contador = 1;
```

```
while (contador <= 10) {
```

```
    imprime("Loop incremental de 1 até " + contador);
```

```
    contador++;
```

```
}
```

```
/*Loop crescente de 1 a 10*/
```

```
var contador = 10;
```

```
while (contador >= 1) {
```

```
    imprime("Loop decremental de 10 até " + contador);
```

```
    contador--;
```

```
}
```

JavaScript: laço de repetição “do while”

JS

```
do {
```

bloco de código a ser executado...

```
} while (condição)
```

```
var contador = 0;  
do {  
    imprime("Este é o número " + contador);  
    contador++;  
} while (contador < 10);
```

JavaScript: break (pausando o loop)

JS

A instrução break "pausa" um loop.

```
/*break*/  
for (var i = 1; i <= 10; i++) {  
    if (i === 3) {  
        imprime("Parou o loop no laço " + i);  
        break;  
    }  
    imprime("Loop normal no laço " + i);  
}
```

JavaScript: break (pausando o loop)

JS

A instrução continue "salta sobre" uma iteração do loop.

```
/*continue*/  
for (var i = 1; i <= 10; i++) {  
    if (i === 3) {  
        continue;  
        imprime("Parou o loop no laço " + i);  
    }  
    imprime("Loop normal no laço " + i);  
}
```

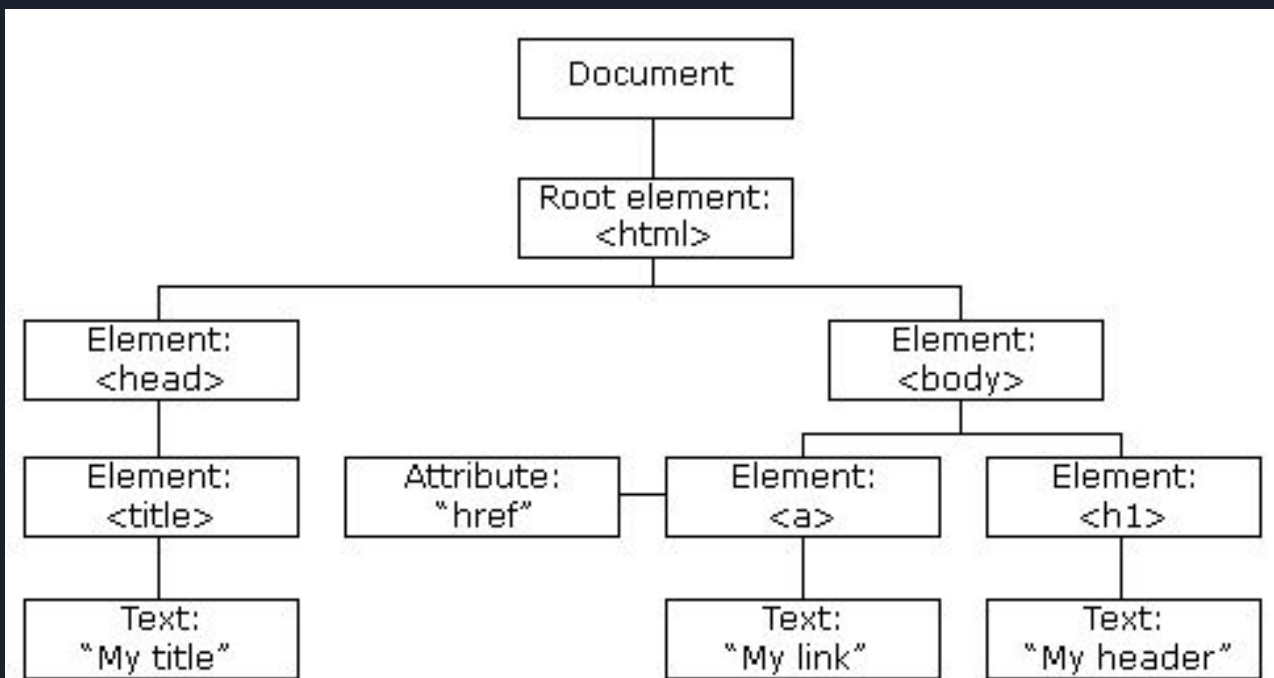
HTML DOM (Document Object Model)



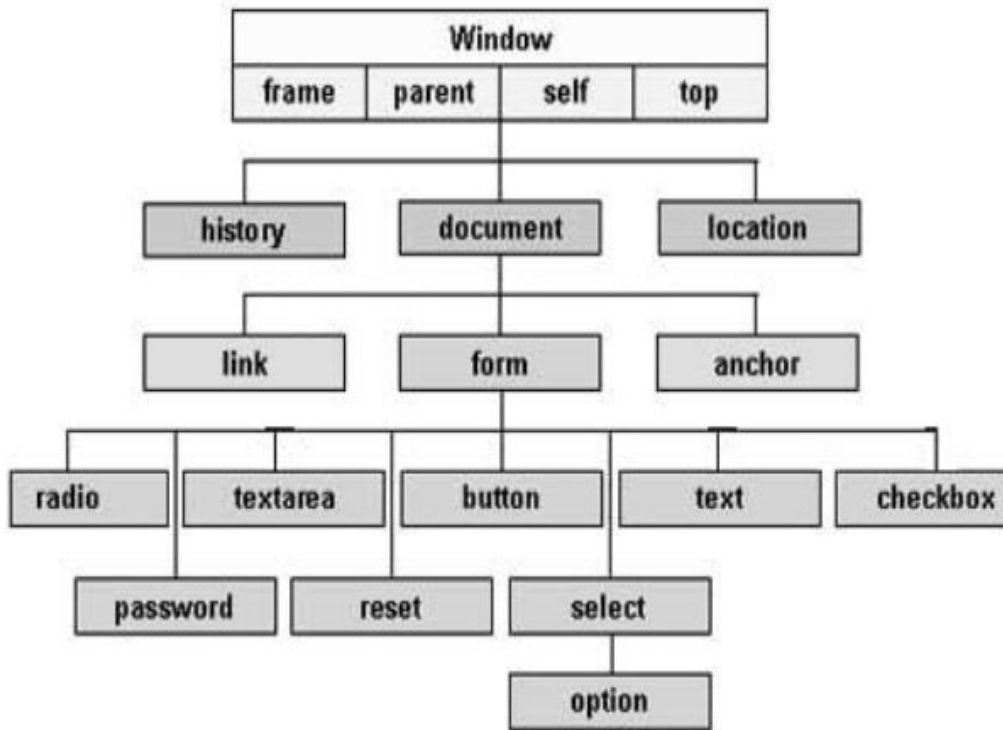
- O DOM é um padrão W3C (*World Wide Web Consortium*).
- Com o DOM HTML, o JavaScript pode acessar e alterar todos os elementos de um documento HTML.
- Quando uma página web é carregado, o navegador cria um **Document Object Modelo** da página.

HTML DOM (Document Object Model)

O HTML DOM modelo é construído como uma árvore de objetos.



HTML DOM (Document Object Model)

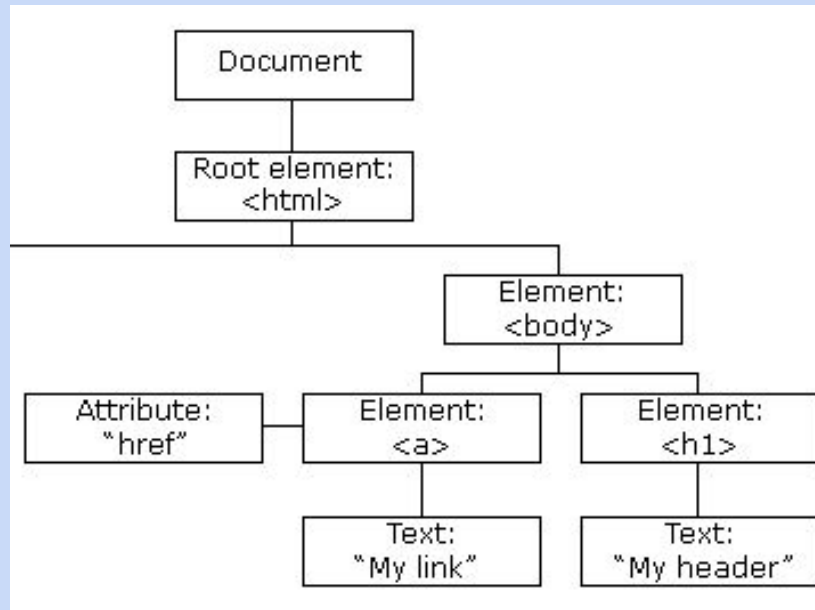


O **DOM** fornece propriedades e métodos para acessar todos os objetos, também chamados de nó, a partir do **JavaScript**.

HTML DOM (Document Object Model)

No HTML DOM, tudo é um nó :

- O documento em si é um nó de **document**.
- Os elementos HTML são nós de elemento.
- Os atributos HTML são nós de atributo.
- Texto dentro elementos são nós de texto.
- Comentários são nós de comentário.



JavaScript: Manipulando HTML com o DOM



Métodos para manipular conteúdo dinamicamente:

Method	Description
<code>document.getElementById(<i>id</i>)</code>	Find an element by element id
<code>document.getElementsByTagName(<i>name</i>)</code>	Find elements by tag name
<code>document.getElementsByClassName(<i>name</i>)</code>	Find elements by class name

JavaScript: Manipulando HTML com o DOM



Mudando elementos HTML

Method	Description
<code>element.innerHTML = new html content</code>	Change the inner HTML of an element
<code>element.attribute = new value</code>	Change the attribute value of an HTML element
<code>element.setAttribute(attribute, value)</code>	Change the attribute value of an HTML element
<code>element.style.property = new style</code>	Change the style of an HTML element

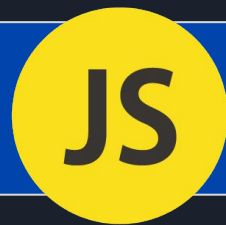
JavaScript: Manipulando HTML com o DOM



Adicionar e eliminar Elements

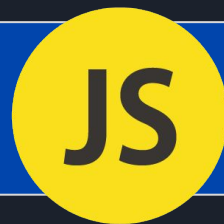
Method	Description
<code>document.createElement(<i>element</i>)</code>	Create an HTML element
<code>document.removeChild(<i>element</i>)</code>	Remove an HTML element
<code>document.appendChild(<i>element</i>)</code>	Add an HTML element
<code>document.replaceChild(<i>element</i>)</code>	Replace an HTML element
<code>document.write(<i>text</i>)</code>	Write into the HTML output stream

Propriedade **innerHTML**



- Útil para **obter** ou **substituir** o conteúdo de elementos HTML.
- Pode ser usado em qualquer elemento HTML, incluindo `<html>` e `<body>`.

O método `getElementById` (obtendo conteúdo)



Fornece o acesso a um elemento HTML, a partir do `id` deste elemento.

```
<p id="meu-elemento">Este é o meu conteúdo</p>

<script>
    var conteudo = document.getElementById("meu-elemento").innerHTML;
    console.log(conteudo);
</script>
//saída no console:
Este é o meu conteúdo
```


O método `getElementById` (inserindo conteúdo)



Fornece o acesso a um elemento HTML, a partir do **id** deste elemento.

```
<p id="meu-elemento"></p>
<script>
    var conteudo =
        document.getElementById("meu-elemento").innerHTML = "meu texto";
</script>
//saída no parágrafo da página html:
meu texto
```

O método `getElementById` (somando conteúdo)



Fornece o acesso a um elemento HTML, a partir do **id** deste elemento.

```
<p id="meu-elemento">Já tem algo aqui!</p>
```

```
<script>
```

```
  var conteudo = document.getElementById("meu-elemento").innerHTML;  
  document.getElementById("meu-elemento").innerHTML = conteudo + " Opa!!";
```

```
</script>
```

//saída no parágrafo da página html:

Já tem algo aqui! Opa!!

O método `getElementsByTagName`

JS

Encontrar elementos HTML a partir do nome da Tag.
Retorna um conjunto de elementos encontrados ou nulo (null).

```
<p>Parágrafo 1</p> <p>Parágrafo 2</p> <p>Parágrafo 3</p> <p>Parágrafo 4</p>  
<script>  
    var todosOsPsDaPagina = document.getElementsByTagName("p");  
    document.write(todosOsPsDaPagina.length);  
</script>  
//saída na página html:  
4
```

O método `getElementsByClassName`

JS

Encontrar elementos HTML a partir do nome de uma classe específica.

```
<p class="dom">1</p>
```

```
<p class="dom">2</p>
```

```
<p>3</p>
```

```
<script>
```

```
    var todosOsPsComClassDom = document.getElementsByClassName("dom");
```

```
    document.write(todosOsPsComClassDom.length);
```

```
</script>
```

```
//saída na página html:
```

```
2
```

O método `querySelector`

JS

Retorna o **Primeiro Elemento** que coincide com um seletor CSS no documento.

```
<p class="dom">1</p>
<p class="dom">2</p>
<script>
    var elemento = document.querySelector(".dom").innerHTML;
    document.write(elemento);
</script>
//saída na página html:
1
```

O método `querySelectorAll`

JS

Retorna **Todos os elementos** que coincidem com um seletor CSS.

```
<p class="dom">1</p>
```

```
<p class="dom">2</p>
```

```
<script>
```

```
    var lista = document.querySelectorAll(".dom");
```

```
    document.write(lista.length);
```

```
</script>
```

```
//saída na página html:
```

```
2
```

Adicionando CSS a um elemento

JS

Para alterar o estilo de um elemento HTML, use esta sintaxe:

```
document.getElementById(id).style.property = new style;
```

```
<p id="dom">1</p>
```

```
<script>
```

```
    var lista = document.getElementById("dom").style.color = "green";
```

```
</script>
```

Inserindo JavaScript no documento HTML

JS

- A partir da tag `<script>` na `<head>` ou na `<body>`:

```
<script>
```

```
document.getElementById("demo").innerHTML = "Olá";
```

```
</script>
```

- JavaScript externo

```
<head>
```

```
<script src="meuScript.js"></script>
```

```
</head>
```


Eventos JavaScript



O DOM HTML permite manipular o código quando ocorre um evento. Os eventos são gerados pelo navegador quando "as coisas acontecem" para elementos HTML:

```
<div class="container">
  <div id="minha-div"></div>
  <button id="butao" class="btn btn-success" onclick="adiciona();">
    Adicionar conteúdo na Div
  </button>
</div>

<script type="text/javascript">
  function adiciona(){
    document.getElementById("minha-div").style.background = "red";
  }
</script>
```

JavaScript: Manipulando HTML com o DOM



```
<p class="intro">The DOM is very useful.</p>
```

```
<p class="intro">This example demonstrates the <b>querySelectorAll</b>  
method.</p>
```

```
<p id="demo"></p>
```

```
<script>
```

```
  var x = document.querySelectorAll("p.intro");
```

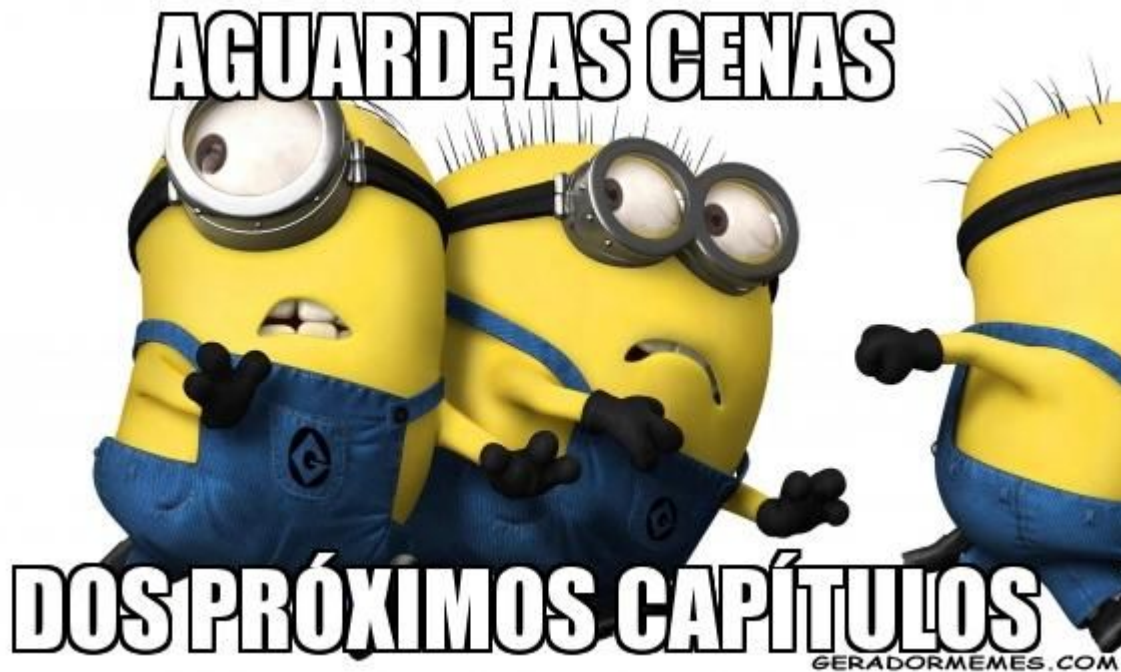
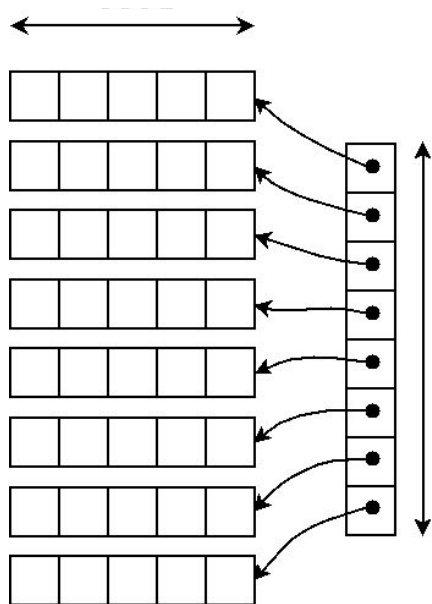
```
  document.getElementById("demo").innerHTML =
```

```
    'The first paragraph (index 0) with class="intro": ' + x[0].innerHTML;
```

```
</script>
```

JavaScript: Arrays e Objetos

JS



JavaScript: Variáveis vs Arrays



Até agora as variáveis nos serviram muito bem, mas...



Arrays em JavaScript

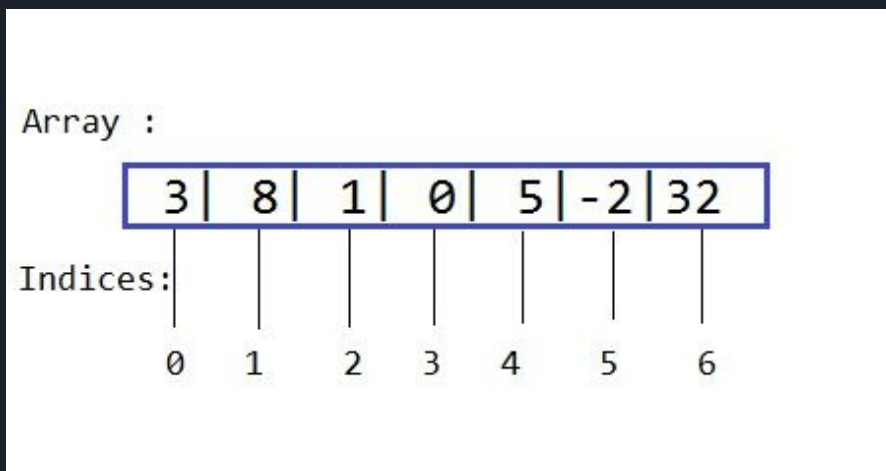


- Em programação de computadores, uma **matriz** (em inglês **array**) é uma estrutura de dados que armazena uma coleção de elementos de tal forma que cada um dos elementos possa ser identificado por, pelo menos, um índice ou uma chave. ([https://pt.wikipedia.org/wiki/Arranjo_\(computação\)](https://pt.wikipedia.org/wiki/Arranjo_(computação))).
- Em uma variável de tipo array, podemos armazenar diversos valores com variados tipos.
- Então, um array é uma variável especial, que pode conter mais de um valor de cada vez.

Arrays em JavaScript



Geralmente, estão estruturados com índices e seus respectivos valores:



Arrays em JavaScript



Declaração:

```
var array_name = [item1, item2, ...];
```

```
var meuArray = [  
  "João",  
  "Maria",  
  "José",  
  "Pedro",  
  "Roberto"  
];
```

```
var meuArray = ["João", "Maria", "José", "Pedro", "Roberto"];
```

Arrays em JavaScript



Tipos diferentes também são permitidos em um mesmo array:

```
var meuArray = [true, "Maria", 123, "Pedro", 15.55];
```



Boolean | String | int | String | float

Arrays em JavaScript



Seus valores serão acessados a partir de seus índices:

```
var meuArray = [true, "Maria", 123, "Pedro", 15.55];
```

```
alert(meuArray[0]); // true  
alert(meuArray[1]); // Maria  
alert(meuArray[2]); // 123  
alert(meuArray[3]); // Pedro  
alert(meuArray[4]); // 15.55
```

```
meuArray[2] = "novo valor"; // substituindo um valor
```

Arrays em JavaScript



Métodos importantes para arrays:

```
var meuArray = ["Maria", 123, "Pedro", 15.55, false];
```

```
console.log(meuArray);
```



```
alert(meuArray.toString()); // retorna: Maria,123,Pedro,15.55,false
```

```
var juntaArray = meuArray.join(" - ");
```

```
alert(juntaArray); // resulta em: Maria - 123 - Pedro - 15.55 - false
```

```
var meuArray = ["Maria", 123, "Pedro", 15.55, false, "Pedro"];
```

```
alert(meuArray.indexOf("Pedro")); // encontra o índice da primeira ocorrência
```

```
alert(meuArray.lastIndexOf("Pedro")); // encontra o índice da última ocorrência
```

```
alert(meuArray.reverse()); // inverte a posição dos elementos
```

Arrays em JavaScript



Métodos importantes para arrays:

```
var meuArray = [true, "Maria", 123, "Pedro", 15.55];

var tamanhoDoArray = meuArray.length; // retorna 5
alert(tamanhoDoArray);
var ordenandoArray = meuArray.sort(); // retorna o array ordenado
alert(ordenandoArray); // imprime: 123,15.55,Maria,Pedro,true

meuArray.push("Programação"); //adiciona um elemento no final
meuArray.pop(); // remove e retorna o último elemento
meuArray.shift(); // remove e retorna o primeiro elemento
meuArray.unshift(); // adiciona um elemento no início
Array.isArray(meuArray); // retorna true se for tipo Array
```



REFERÊNCIAS:

LACERDA, Ivan Max Freire de, OLIVEIRA, Ana Liz Souto.

Programador Web: um guia para programação e manipulação de banco de dados.

Rio de Janeiro: Senac Nacional, 2013.

W3schools: <http://www.w3schools.com>

Slide: Algoritmo e lógica de programação:

https://pt.slideshare.net/engenhariadecomputacao/algoritmo-e-logica-de-programao-aula-1?qid=4b7e1bbc-c2c4-4e0e-b267-e50d0867c402&v=&b=&from_search=2

Slide: Material de apoio de algoritmo e lógica de programação:

https://pt.slideshare.net/rodfernandes/material-de-apoio-de-algoritmo-e-logica-de-programao?qid=d0e7d40-1d80-4a6b-a730-d5da859fe9f1&v=&b=&from_search=1

Apontamentos da Aula:

<https://github.com/adrielsales/senac/wiki/Aulas-Senac>