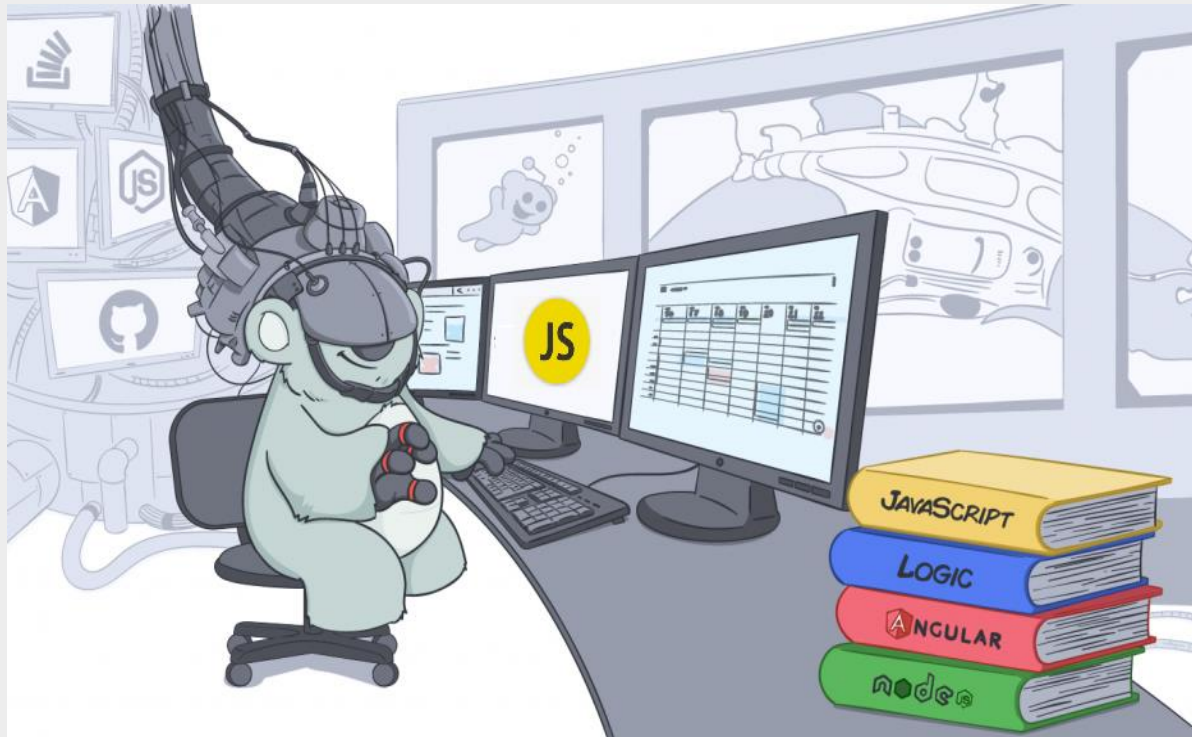


Desenvolvimento Web

Unidade 2 – Parte 1

Introdução ao Javascript



Prof. Aparecido V. de Freitas
Doutor em Engenharia
da Computação pela EPUSP
aparecido.freitas@online.uscs.edu.br
aparecidovfreitas@gmail.com

Bibliografia



Ambiente Javascript

Para se criar programas executáveis em um computador precisa-se de uma linguagem de Programação;

Neste curso utilizaremos a linguagem **Javascript**.



Javascript

- ✓ Linguagem tipicamente interpretada;
- ✓ Código Javascript pode ser executado sob um browser.

```

let subjectAverage = 0;
query(
  "SELECT * FROM marks WHERE subject_ID=" + subject_ID
)
function (datasetsWithSubject) {
  if (datasetsWithSubject.length > 0) {
    subjectAverage = 0;
    datasetsWithSubjectLength = datasetsWithSubject.length;
    datasetsWithSubject.forEach((dataset) => {
      subjectAverage += parseFloat(dataset[0]);
    });
    subjectAverage = subjectAverage / datasetsWithSubjectLength;
  } else {
    subjectAverage = 0;
  }
}

```


Como um browser pode processar código Javascript?



Javascript

- ✓ Um browser geralmente possui um motor capaz de processar o código Javascript embutido numa página HTML.

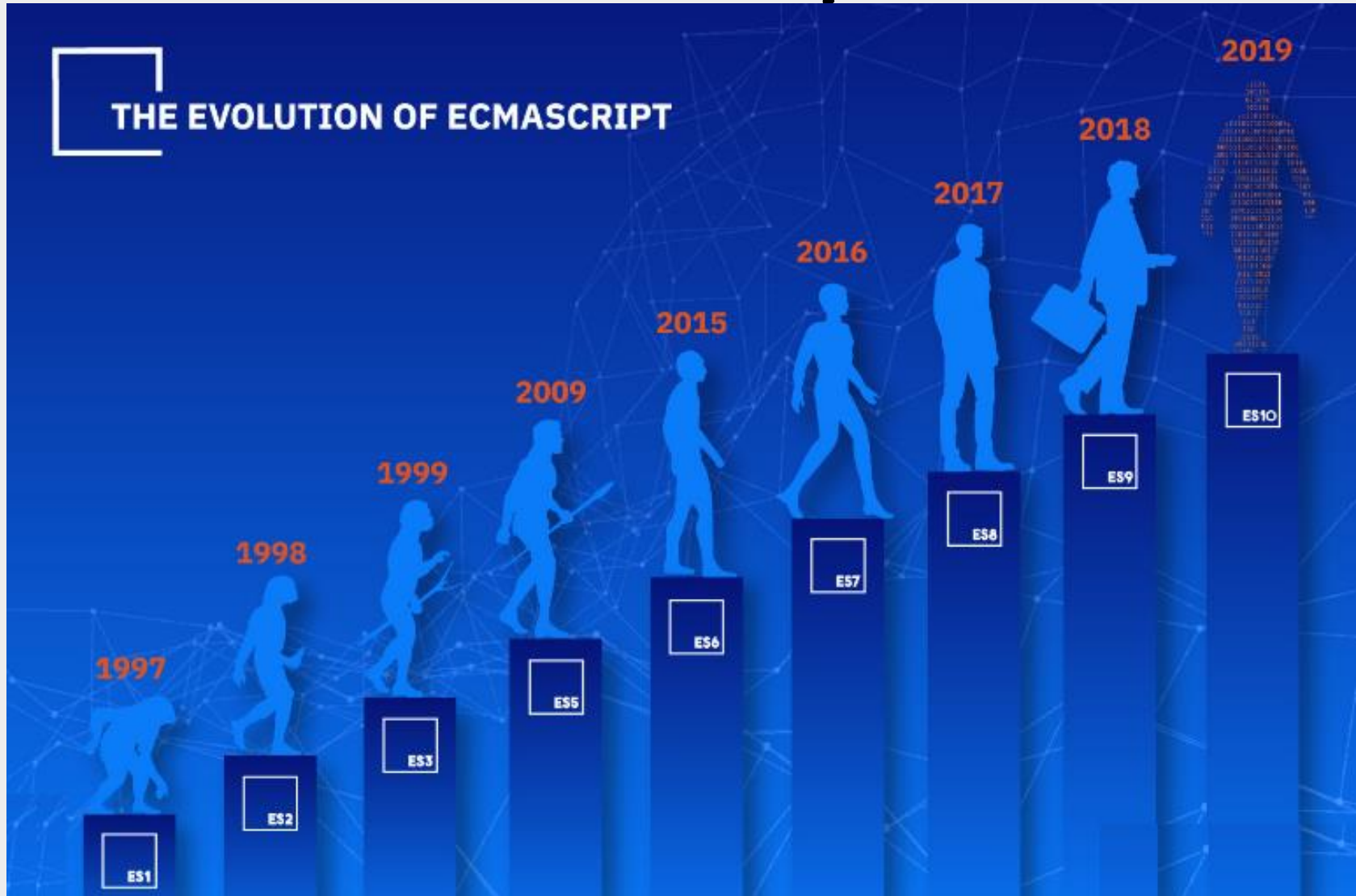


Javascript

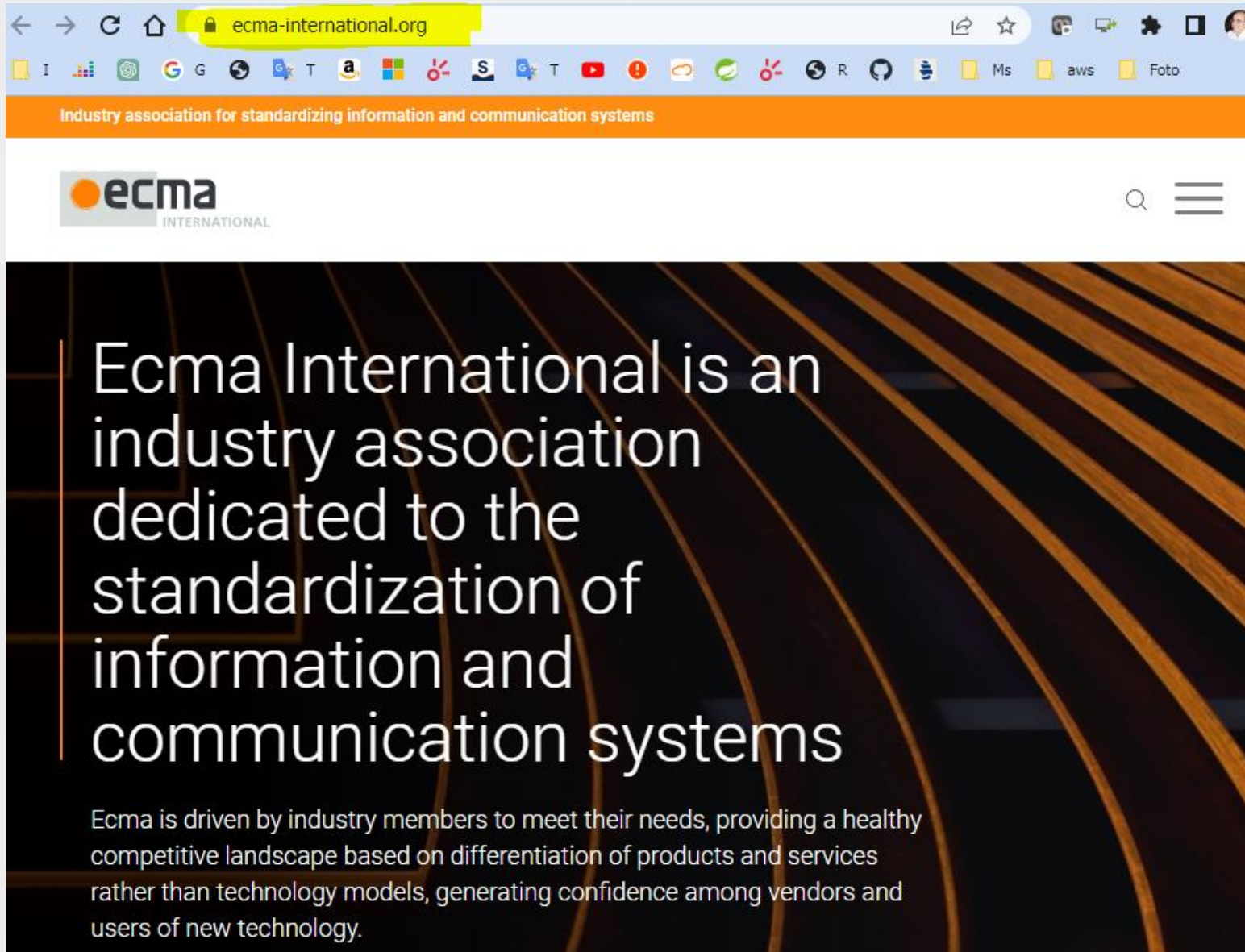
- ✓ Desenvolvida pela Netscape em conjunto com a Sun;
- ✓ 1995 – primeira versão;
- ✓ 1997 – linguagem administrada pela ECMA;
- ✓ ECMA – European Computer Manufacturers Association;
- ✓ A linguagem também é conhecida por ECMAScript



Javascript









<https://www.ecma-international.org/>



The screenshot shows the homepage of the Ecma International website. The browser's address bar displays "ecma-international.org". Below the browser window, an orange banner reads "Industry association for standardizing information and communication systems". The main header features the "ecma INTERNATIONAL" logo on the left and a search icon with a menu icon on the right. The central content area has a dark background with a pattern of curved, golden-brown lines. Large white text reads: "Ecma International is an industry association dedicated to the standardization of information and communication systems". Below this, smaller white text states: "Ecma is driven by industry members to meet their needs, providing a healthy competitive landscape based on differentiation of products and services rather than technology models, generating confidence among vendors and users of new technology."

[Home](#) » [TIOBE Index](#)

TIOBE Index for September 2025

Sep 2025	Sep 2024	Change	Programming Language	
1	1			Python
2	2			C++
3	4	▲		C
4	3	▼		Java
5	5			C#
6	6			JavaScript

O que fazer com Javascript ?



```
document.getElementById(div).innerHTML = "Email inválido";  
else if (i==2)  
    message = errors[1];  
    divs[1];  
    var atpos=inputs[i].indexOf("@");  
    var dotpos=inputs[i].lastIndexOf(".");  
    if (atpos<1 || dotpos<atpos+2 || dotpos>inputs[i].length-2 ||  
        document.getElementById('errEmail').innerHTML = "Email inválido";  
    else  
        document.getElementById(div).innerHTML = "Email válido";
```

Javascript ?

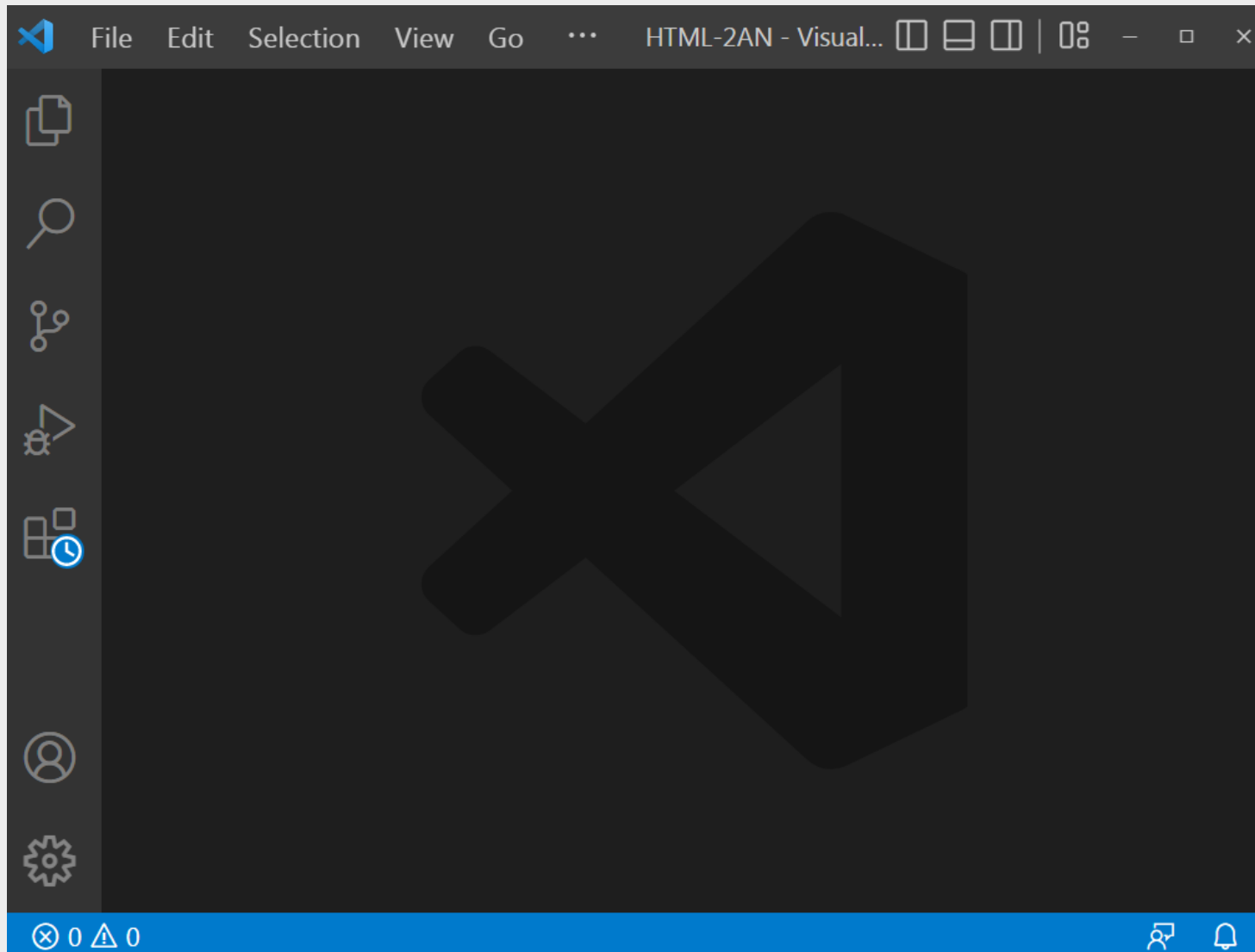
- ✓ Largamente utilizada em sistemas Web;
- ✓ Atua em conjunto com HTML (estrutura da página) e CSS (estilos);
- ✓ Permite interação dos usuários às páginas (forms);
- ✓ Implementam dinamismo às páginas.





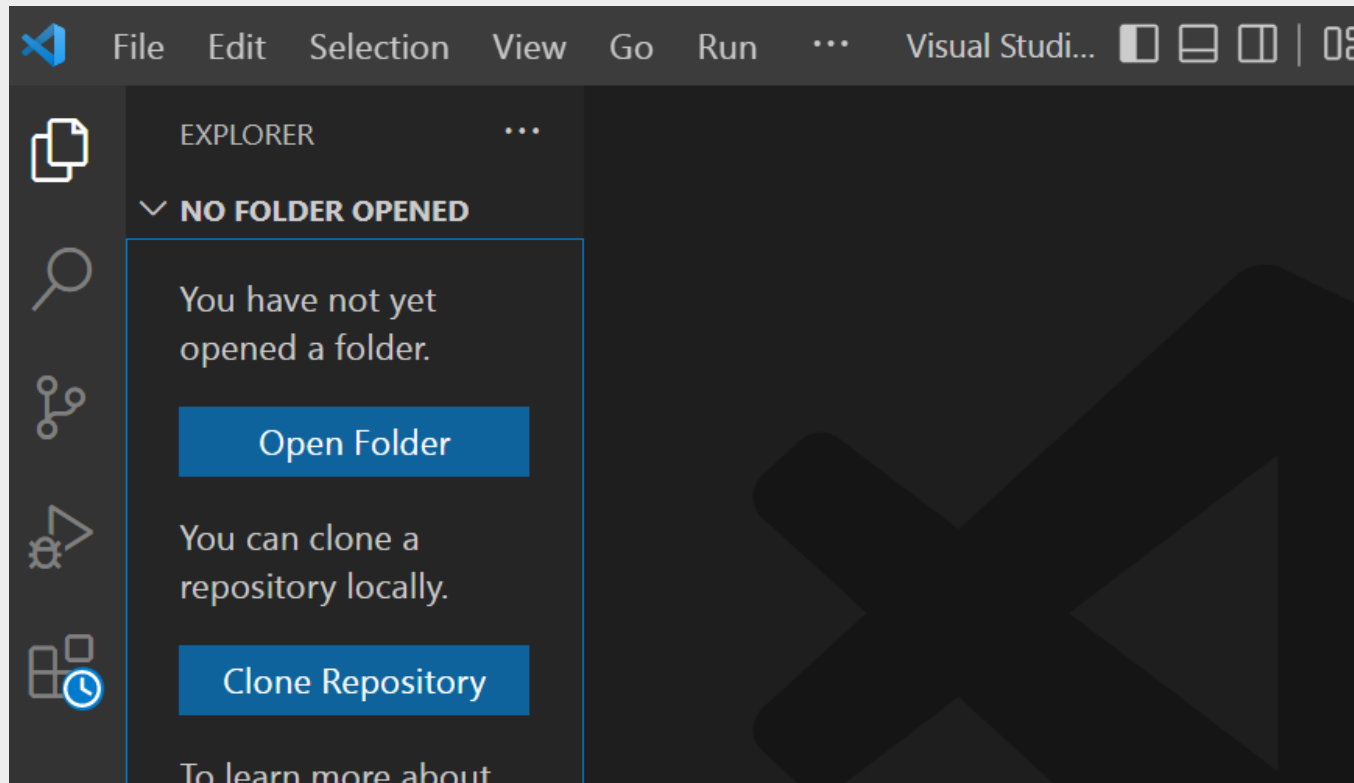
Como escrever código JS ?

VS Code



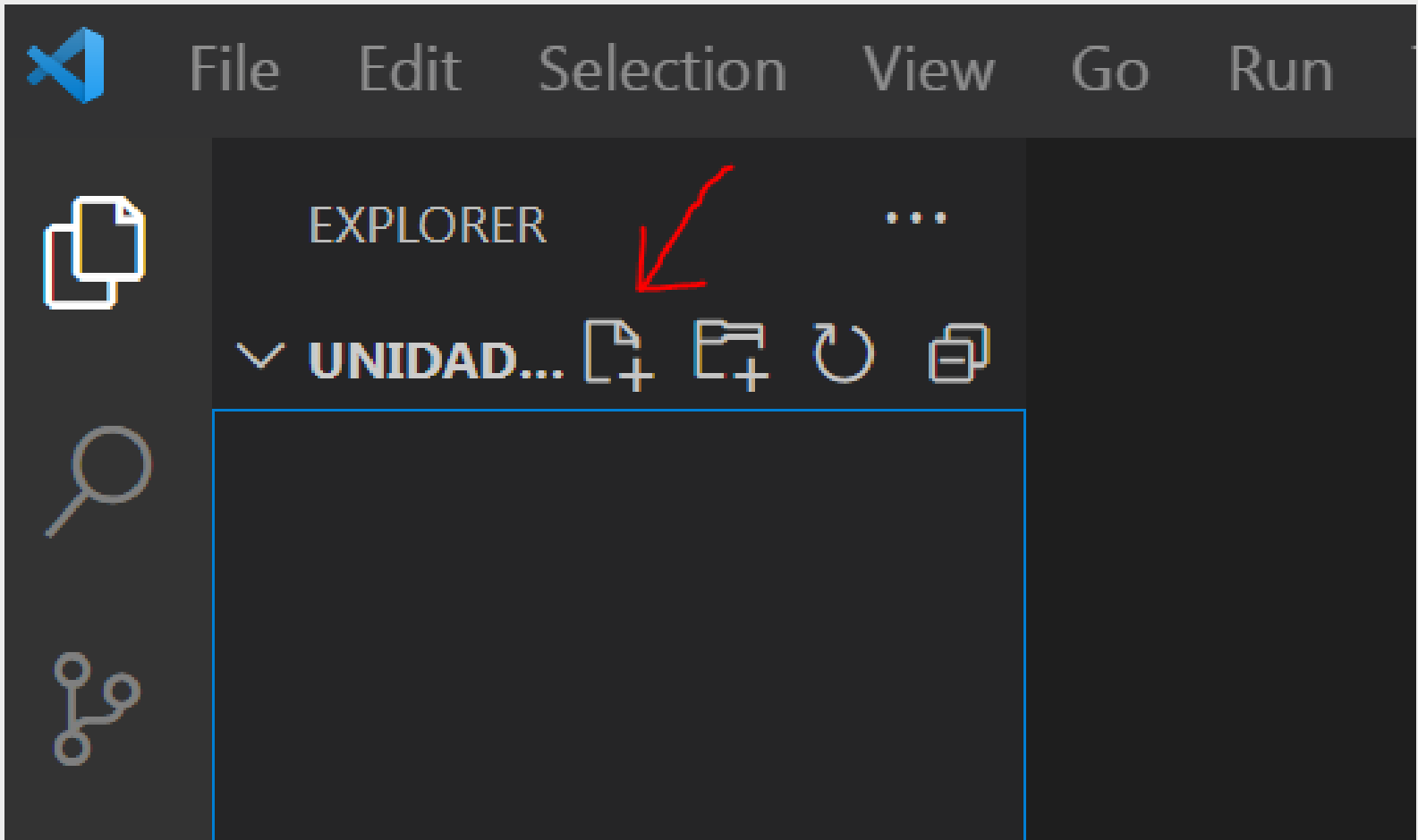
Javascript – Hello World!

- ✓ Crie a pasta onde o código Javascript será criado;
- ✓ Abra o Visual Studio Code;
- ✓ Tecle Ctrl + Shift + E
- ✓ Defina o folder.



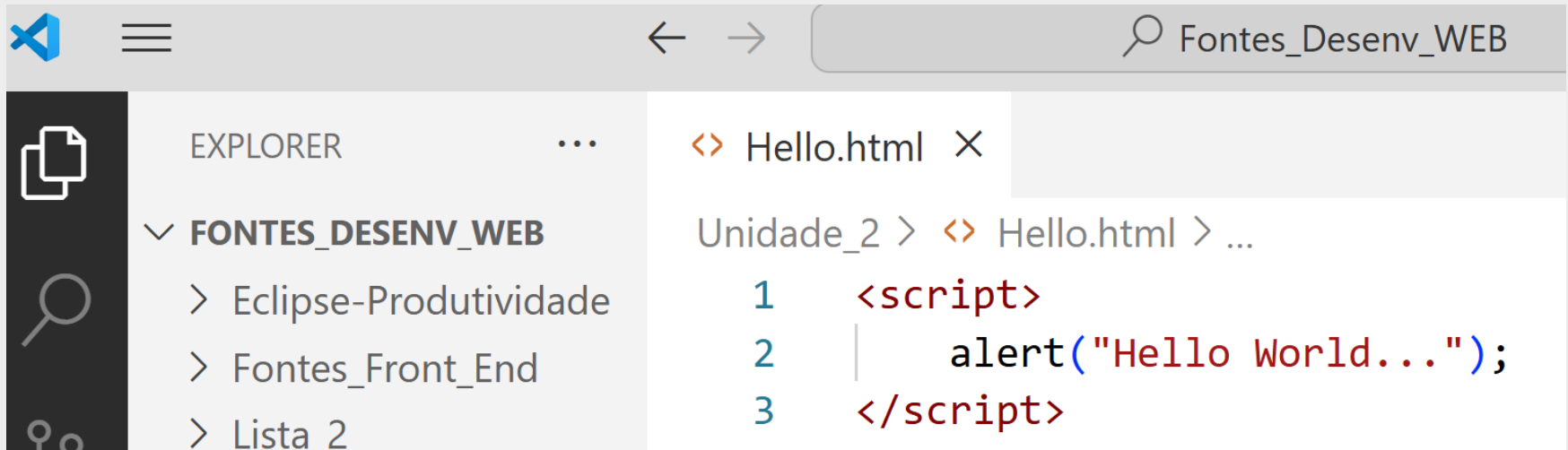
Javascript – Hello World!

- ✓ Criação do arquivo hello.html



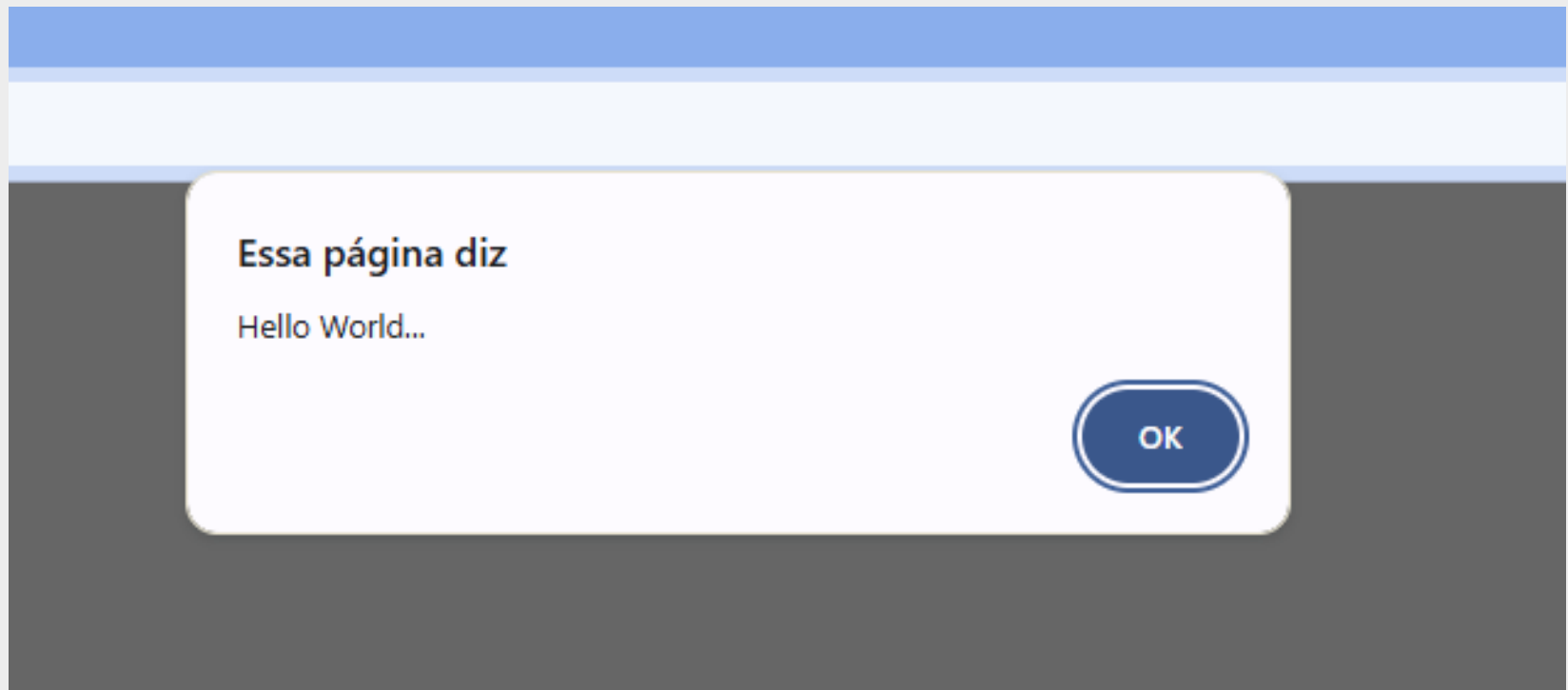
Javascript – Hello World!

✓ Criação do arquivo Hello.html



Javascript – Hello World!

- ✓ Abra o browser e digite na barra de endereços:
C:/Unidade_2/Hello.html



console.log()

- ✓ A função `console.log()` é usada na linguagem JavaScript para exibir mensagens ou valores no console do navegador ou no ambiente em que o código está sendo executado.
- ✓ É uma ferramenta de depuração útil para os desenvolvedores, pois permite que eles acompanhem o comportamento do código e visualizem informações relevantes durante a execução do programa.

javascript

 Copy code

```
console.log("Olá, mundo!"); // Exibe a mensagem "Olá, mundo!"
```

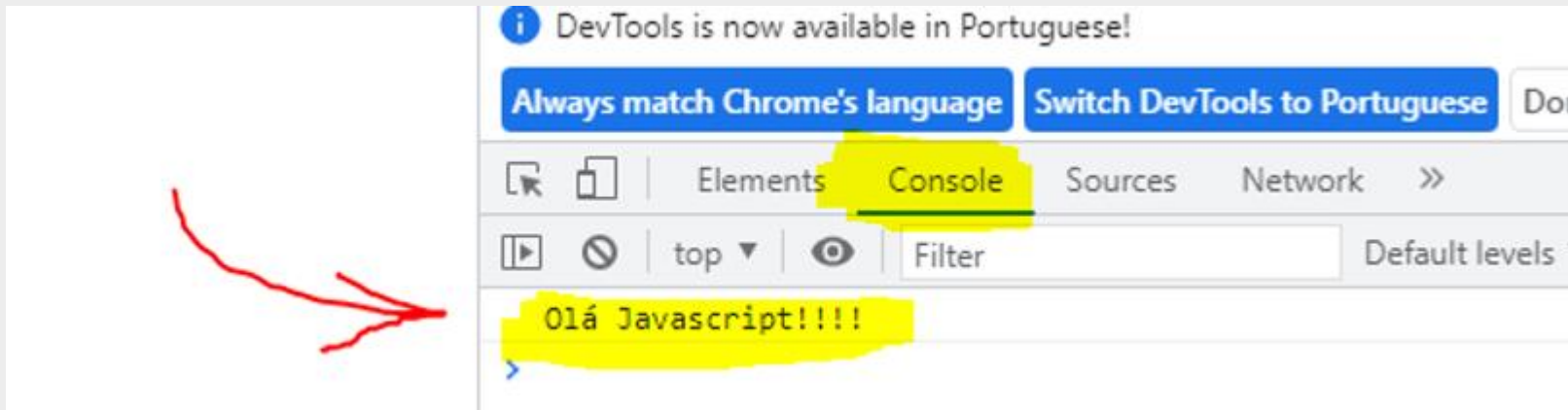
- ✓ A informação especificada pelo `console.log` pode ser visualizada por meio de ferramentas de desenvolvimento do navegador;
- ✓ `console.log()` é particularmente útil para depurar código e verificar o valor de variáveis em diferentes pontos do programa.

Javascript – console.log()

```
1  <script>  
2  |   console.log("Olá Javascript!!!!");  
3  </script>
```

Javascript – console.log()

- ✓ Abra o browser e execute o código;
- ✓ No Google Chrome tecle **F12** (ou **ctrl shift I**) para acessar as ferramentas do desenvolvedor e clique na aba Console.





Tarefa

- ✓ Por meio da IDE VSCode escreva um código Javascript para exibir em tela e na console do navegador o seu nome completo.
- ✓ Adicionalmente, por meio da IDE VSCode escreva um código JS para exibir em tela e na console do navegador o seu endereço.
- ✓ Adicionalmente, por meio da IDE VSCode escreva um código JS para exibir em tela e na console do navegador o seu e-mail.

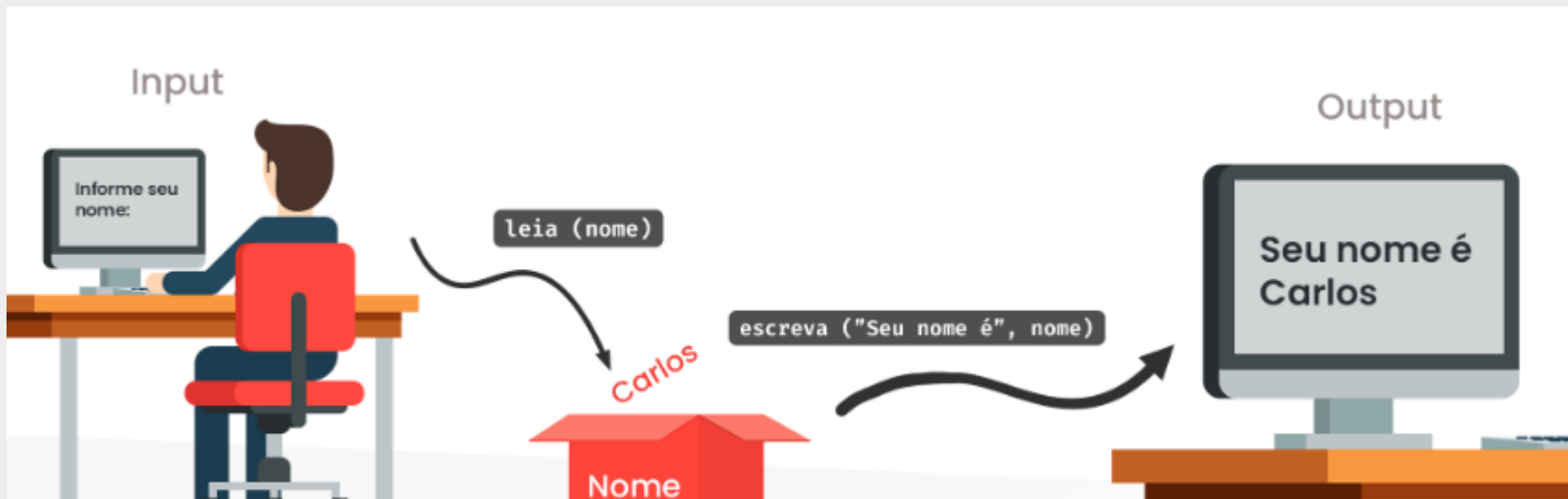
Programa Javascript

- @ A grande maioria dos programas de computador faz **interação** com o usuário;
- @ Assim, o usuário entra com informações que devem ser processadas pelo programa e respostas são devolvidas ao usuário.



Programa Javascript

- @ A maioria dos programas efetua diversas operações com as informações fornecidas pelo usuário;
- @ Cada operação pode produzir resultados **intermediários**, os quais serão empregados em outras operações subsequentes.



Como essas informações são manipuladas pelo programa ?



Variáveis

Permitem que dados fornecidos pelo usuário ou criados em tempo de processamento sejam armazenados e manipulados pelo programa em tempo de execução.



Declarando variáveis

- ✓ Na maioria das linguagens de programação, uma variável deve ser declarada antes de ser usada. Javascript não é exceção;
- ✓ Declarar uma variável é simplesmente "reservar" o nome da variável;
- ✓ Em JavaScript, nomes de variáveis podem consistir de quaisquer sequências de letras (maiúsculas ou minúsculas), dígitos, underscore, sinal de dólar, mas não devem iniciar com dígito.

```
var isTrue = true;
if(isTrue === true){
  console.log('Variavel é verdadeira!')
} else {
  console.log('Variavel é falsa!')
}
```

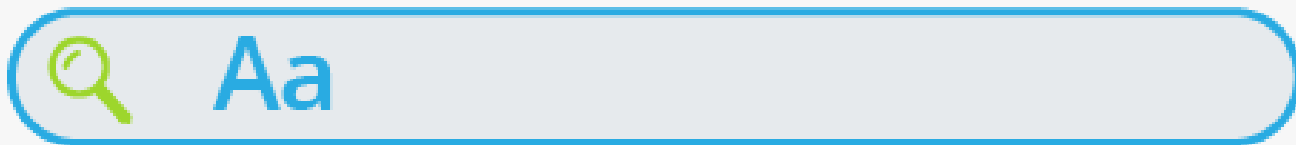

Palavras reservadas

- ✓ Existem nomes que são palavras reservadas e, portanto, não podem ser usadas para se declarar variáveis.

abstract	arguments	await	boolean
break	byte	case	catch
char	class	const	continue
debugger	default	delete	do
double	else	enum	eval
export	extends	false	final
finally	float	for	function
goto	implements	if	import
in	instanceof	int	interface
let	long	native	new
null	package	private	protected
public	return	short	static
super	switch	synchronized	this
throw	throws	transient	true
try	typeof	var	void
volatile	while	with	yield

Variáveis Javascript

- ✓ São sensitive-case;
- ✓ Maiúsculo é diferente de minúsculo.



Como declarar variáveis ?



Variáveis Javascript

✓ São declaradas por meio de:

- **var**
- **let**
- **const**

Variáveis Javascript

✓ São declaradas por meio de:

- **var**
- **let**
- **const**

var e let

- ✓ Existem diferenças técnicas relacionadas a escopo entre var e let;
- ✓ Use let na maioria dos casos (mais seguro, escopo de bloco, evita erros);
- ✓ Use var só se precisar de compatibilidade com código antigo.

let e const

◆ Diferenças entre `let` e `const`

Característica	<code>let</code>	<code>const</code>
Escopo	Bloco <code>{ }</code>	Bloco <code>{ }</code>
Redeclaração no mesmo escopo	✗ Não permitido	✗ Não permitido
Reatribuição	✓ Permitida	✗ Proibida
Inicialização obrigatória	✗ Não precisa inicializar na declaração	✓ Deve ser inicializada na declaração

Declarando Variáveis

```
1  <script>
2
3      let A;
4      console.log(A);
5      console.log(B);
6
7  </script>
```

Declarando Variáveis

```
1  <script>
2
3      let A;
4      console.log(A);
5      console.log(B);
6
7  </script>
```

✖ Uncaught ReferenceError: B is not defined

- ✓ Variável **A** foi declarada mas não inicializada!
- ✓ Variável **B** não foi declarada !!!

Declarando Variáveis

- ✓ Variáveis declaradas com **let** não podem ser redeclaradas.

```
1  <script>
2
3      let A;
4      let A;
5      console.log(A);
6
7  </script>
```

✖ Uncaught SyntaxError: Identifier
'A' has already been declared

Comandos de atribuição

- ✓ Define ou redefine o valor armazenado no local de armazenamento indicado por uma variável;
- ✓ A instrução de atribuição muitas vezes permite que o mesmo nome de variável possa conter valores diferentes em momentos diferentes durante a execução do programa;
- ✓ Em **JavaScript**, a atribuição de valor para uma variável é feita com o sinal "=".

```
let a = 10;  
  
a = a + 5;
```

Inicializando variáveis

- ✓ Após uma declaração bem sucedida, a variável deve ser inicializada;
- ✓ Para isso, usa-se o comando de atribuição.

Inicializando variáveis

```

1  <script>
2
3      let A = 180;
4      let B = A;
5      let C;
6      console.log(A); // -> 180
7      console.log(B); // -> 180
8      C = 70;
9      console.log(C); // -> 70
10
11 </script>
  
```

Operadores aritméticos

```

1
2  <meta charset="UTF-8">
3  <script>
4      const x = 5;
5      const y = 2;
6
7      console.log("adição: ", x + y);           // -> 7
8      console.log("subtração: ", x - y);        // -> 3
9      console.log("multiplicação: ", x * y);     // -> 10
10     console.log("divisão: ", x / y);           // -> 2.5
11     console.log("resto da divisão:", x % y);    // -> 1
12     console.log("exponenciação: ", x ** y);    // -> 25
13 </script>
  
```


Operadores aritméticos unários

```

1
2  <meta charset="UTF-8">
3  <script>
4      let str = "123";
5      let n1 = +str;
6      let n2 = -str;
7      let n3 = -n2;
8      let n4 = +"abcd";
9
10     console.log(`${str} : ${typeof str}`); // -> 123 : string
11     console.log(`${n1} : ${typeof n1}`);   // -> 123 : number
12     console.log(`${n2} : ${typeof n2}`);   // -> -123 : number
13     console.log(`${n3} : ${typeof n3}`);   // -> 123 : number
14     console.log(`${n4} : ${typeof n4}`);   // -> NaN : number
15  </script>

```

Operadores ++ e --

```
1
2 <meta charset="UTF-8">
3 <script>
4     let n1 = 10;
5     let n2 = 10;
6
7     console.log(n1);           // -> 10
8     console.log(n1++);        // -> 10
9     console.log(n1);           // -> 11
10
11    console.log(n2);           // -> 10
12    console.log(++n2);         // -> 11
13    console.log(n2);           // -> 11
14
15    let n3 = 20;
16    let n4 = 20;
17
18    console.log(n3);           // -> 20
19    console.log(n3--);         // -> 20
20    console.log(n3);           // -> 19
21
22    console.log(n4);           // -> 20
23    console.log(--n4);          // -> 19
24    console.log(n4);           // -> 19
25 </script>
```

Atribuição Composta

```
1
2  <meta charset="UTF-8">
3  <script>
4      let x = 10;
5
6      x += 2;
7      console.log(x); // -> 12
8      x -= 4;
9      console.log(x); // -> 8
10     x *= 3;
11     console.log(x); // -> 24
12     x /= 6;
13     console.log(x); // -> 4
14     x **= 3;
15     console.log(x); // -> 64
16     x %= 10;
17     console.log(x); // -> 4
18 </script>
```

Operadores Lógicos

- ✓ Operam com valores booleanos **true** ou **false**;
- ✓ Retornam valores somente deste tipo;
- ✓ **Javascript** provê três operadores:

AND	=>	&&
OR	=>	
NOT	=>	!

Operadores Lógicos

```

1
2  <meta charset="UTF-8">
3  <script>
4      console.log(!true);           // -> false
5      console.log(!false);          // -> true
6
7      const a = false;
8      const b = true;
9      const c = false;
10     const d = true;
11
12     console.log(a && b && c || d);    // -> true
13     console.log(a && b && (c || d));  // -> false
14 </script>
  
```

Modificando variáveis

```
1  <script>
2
3      "use script";
4      let A = 5;
5      console.log(A);           // -> 5
6      A = 7;                   // -> 7
7      console.log(A);
8      A = A + 4;
9      console.log(A);           // -> 11
10
11
12 </script>
```




Javascript é uma linguagem dinamicamente tipada!

O que significa uma linguagem ser dinamicamente tipada ?



Linguagem dinamicamente tipada

- ✓ Uma linguagem de programação ser dinamicamente tipada significa que as variáveis não possuem um tipo de dado fixo e podem ser alteradas em tempo de execução;
- ✓ Em outras palavras, o tipo de uma variável é determinado em tempo de execução, conforme os valores a ela atribuídos.

```

query(
  "SELECT * FROM marks WHERE subject = ?"
)
function (datasetsWithSubject) {
  if (datasetsWithSubject.length > 0) {
    subjectAverage = 0;
    datasetsWithSubjectLength = datasetsWithSubject.length;
    datasetsWithSubject.forEach((dataset) => {
      subjectAverage += parseFloat(dataset.subjectAverage);
    });
  }
}

```

Linguagem dinamicamente tipada

- ✓ Nas linguagens de programação dinamicamente tipadas, você não precisa declarar explicitamente o tipo de uma variável ao criar ou atribuir um valor a ela.
- ✓ A verificação de tipo ocorre em tempo de execução, e não em tempo de compilação.



Linguagem dinamicamente tipada

```

1  <script>
2
3      "use script";
4
5  → let A = "USCS"
6      console.log(A); // -> USCS
7
8  → A = 9;
9      console.log(A); // -> 9
10
11 </script>
  
```

Observação

Além de ser dinamicamente tipada, Javascript vai um passo adiante, pois além de permitir mudança dos tipos de dados em variáveis, também executa conversão implícita de dados, caso seja necessário!



```

28 var ss_legacy = function(node) {
29
30     if (!node instanceof Object) return false;
31
32     if (node.length) {
33         for (var i=0; i<node.length; i++) {
34             ss_legacy(node[i]);
35         }
36         return;
37     };
38
39     if (node.value) {
40         node.value = ss_liga(node.value);
41     } else if (node.nodeValue) {
42         node.nodeValue = ss_liga(node.nodeValue);
43     } else if (node.innerHTML) {
44         node.innerHTML = ss_liga(node.innerHTML);
45     }
46
47 };
48
49 var ss_getElementsByClassName = function(node, classname) {
50     var a = [];
51     var re = new RegExp('(' + classname + '|' + '|$)');
52     var els = node.getElementsByTagName("*");
53     for(var i=0, j=els.length; i<j; i++)
54         if(re.test(els[i].className))a.push(els[i]);
55     return a;
56 };
57
58 var ss_liga = function(that) {
59     var re = new RegExp(ss_keywords.join('|').replace(/[-[\]{}()*+?.,\\$%&'"/g, "\\$&"), "gi");
60     return that.replace(re, function(v) {
61         return ss_icons[v.toLowerCase()];
62     });
63 };

```

Conversão implícita

```
1  <script>
2
3      "use script";
4
5      let contador = 8;
6      let A = "USCS!";
7
8      A = contador + A;
9      console.log(A);           // -> 8USCS!
10
11 </script>
```


Constantes

- ✓ A keyword `const` é usada para declarar constantes;
- ✓ Constantes também armazenam valores de dados, mas uma vez inicializados, seus valores não podem mais serem modificados.

```
1  <script>
2
3      "use script";
4
5      const A = "USCS";
6
7      A = "Computação";
8
9      console.log(A);
10
11 </script>
```


Constantes

```
1  <script>
2
3      "use script";
4
5      const A = "USCS";
6
7      A = "Computação";
8
9      console.log(A);
10
11 </script>
```



✘ Uncaught TypeError: Assignment to constant variable.

Comentários em Javascript

```

1
2  <meta charset="UTF-8">
3  <script>
4
5      // Comentário de linha única.
6      let A = 9; // Variável A recebe o valor 9.
7
8      /*
9          Comentário com diversas linhas
10         Pode se estender por
11         várias linhas de código.
12     */
13     let B = 5; // Variável B recebe o valor 5.
14
15 </script>
  
```

Tipos de Dados

- ✓ Em JavaScript, os tipos de dados são divididos em primitivos (ou simples) e complexos (ou compostos);
- ✓ Entre os tipos primitivos, temos os números, strings de caracteres e booleanos;
- ✓ Entre os tipos complexos, temos, por exemplo, arrays e objetos.



Operador typeof

- ✓ Operador unário (possui apenas um argumento);
- ✓ Retorna um string que corresponde ao tipo do dado passado como argumento;
- ✓ O argumento pode ser um literal ou uma variável;

```
4  typeof("Hola") // string
5  typeof(12) // integer
6  typeof(true) // boolean
7  typeof(undefined) // undefined
8  typeof(null) // object
9  typeof({}) // object
10 typeof(Symbol()) // symbol
11 typeof(1n) // bigint
```



Operador typeof

```
2  <meta charset="UTF-8">
3  <script>
4
5      let ano = 2023;
6      console.log(typeof ano);           // -> number
7      console.log(typeof 2023);          // -> number
8
9      let universidade = "USCS";
10     console.log(typeof universidade);   // -> string
11     console.log(typeof "USCS");         // -> string
12
13     let tipoAno = typeof ano;
14     console.log(tipoAno);                // -> number
15     console.log(typeof tipoAno);        // -> string
16
17 </script>
```

Tipos Primitivos

- ✓ Em **JavaScript**, há seis tipos de dados primitivos:

"undefined"

"boolean"

"number"

"bigint"

"string"

"symbol"

boolean

- ✓ Pode conter apenas um de dois valores: **true** ou **false**.

```
1
2  <meta charset="UTF-8">
3  <script>
4      let A = true;
5      let B = false;
6
7      console.log(B);           // -> false
8      console.log(typeof false); // -> boolean
9      console.log(A);           // -> true
10     console.log(typeof A);     // -> boolean
11
12 </script>
```


number

- ✓ Representa tanto números reais quanto inteiros.

```

1
2 <meta charset="UTF-8">
3 <script>
4     const raio = 10;
5     let pi = 3.1416;
6     let area = (raio * raio * pi);
7     let metadeArea = area / 2;
8
9     console.log(raio);                      // -> 10;
10    console.log("Área = " + area.toFixed(2)); // -> 314.16
11    console.log("Metade da área = " + metadeArea.toFixed(2)); // -> 157.08
12    console.log(typeof raio);                // -> number
13    console.log(typeof area);                // -> number
14    console.log(typeof metadeArea);          // -> number
15 </script>
  
```


number – valores especiais

- ✓ Infinity, -Infinity e NaN (not a number);
- ✓ Os dois primeiros correspondem exatamente ao que sabemos da Matemática;
- ✓ O último, NaN, corresponde a uma notificação de que alguma operação matemática não pode ser executada.



number – valores especiais

```
1
2  <meta charset="UTF-8">
3  <script>
4
5      let X = 99 / 0;
6      let Y  = -Infinity;
7
8      console.log(X);           // -> Infinity
9      console.log(Y);           // -> -Infinity
10     console.log(typeof X);     // -> number
11     console.log(typeof Y);     // -> number
12
13     let U = "USCS";
14     let W = U * 5;
15     console.log(W);             // -> NaN
16     console.log(typeof W);     // -> number
17
18  </script>
```

bigint

- ✓ Não são usados com frequência;
- ✓ Permite manipularmos números muito grandes;
- ✓ Literais **bigint** são representados com o sufixo **n**.

```

1
2  <meta charset="UTF-8">
3  <script>
4
5      let big = 123456789000000000000000n;
6      let big2 = 4n;
7
8      console.log(big);           // -> 123456789000000000000000n
9      console.log(typeof big);    // -> bigint
10
11     console.log(big2);           // -> 4n
12
13  </script>

```

string

- ✓ Representam uma sequência de caracteres formando texto;
- ✓ São imutáveis. Para se alterar um caractere do string, deve-se criar um novo string;
- ✓ Literais string podem ser definidos com " ou '.



string

```

1
2  <meta charset="UTF-8">
3  <script>
4
5      let nome = "Carlos";
6      let sobrenome = 'Silva';
7
8      console.log(nome);           // -> Carlos
9      console.log(typeof nome);    // -> string
10     console.log(sobrenome);       // -> Silva
11     console.log(typeof sobrenome); // -> string
12
13  </script>
  
```


Operações com strings

- ✓ As operações aritméticas de subtração, multiplicação ou divisão com strings retornam erro. Mais precisamente retornam NaN;
- ✓ A exceção é a adição, pois a operação será tratada como concatenação de strings.



Operações com strings

```

1
2 <meta charset="UTF-8">
3 <script>
4
5     let path = "C:\\Windows" - "Windows";
6     console.log(path);                // -> NaN
7
8     let test = "100" - "10";
9     console.log(test);                // -> 90
10    console.log(typeof test);         // -> number
11
12    path = "C:\\\\" + "Windows";
13    console.log(path);                // -> C:\Windows
14
15    test = "100" + "10";
16    console.log(test);                // -> 10010
17    console.log(typeof test);         // -> string
18
19 </script>
  
```

Interpolação de strings

```
1
2 <meta charset="UTF-8">
3 <script>
4
5     let cidade = "São Caetano do Sul";
6     let estado = "SP";
7
8     let frase = `${cidade} está localizada em ${estado}.`;
9     console.log(frase); // -> São Caetano do Sul está localizada em SP.
10
11 </script>
```


undefined

- ✓ Corresponde ao valor default que todas variáveis possuem caso após suas declarações não tenham algum valor à elas atribuído;
- ✓ Caso queiramos que uma variável tenha algum valor sem significado, deveremos atribuir a ela o valor **null**.

undefined

```

1
2  <meta charset="UTF-8">
3  <script>
4
5  let A;
6  console.log(typeof A);      // -> undefined
7
8  A = 5;
9  console.log(typeof A);      // -> number
10
11 A = null;
12 console.log(typeof A);      // -> object
13
14 console.log(typeof B);      // -> undefined
15 console.log(B);             // -> Uncaught ReferenceError
16
17
18 </script>

```

Criação de dados com Construtores

```
<meta charset="UTF-8">
<script>
    const texto = String();
    const valor = Number();
    const booleano = Boolean();

    console.log(texto);      // -> ""
    console.log(valor);      // -> 0
    console.log(booleano);   // -> false

    const big1 = BigInt(99);
    console.log(big1);       // -> 99n
</script>
```

Conversão de tipos

- ✓ Conversões em **Javascript** ocorrem de forma automática;
- ✓ Mas, pode-se também utilizar funções construtoras que aceitam argumentos e os convertem para outros tipos de dados.

Conversão de tipos

```

1
2  <meta charset="UTF-8">
3  <script>
4
5      const num = 99;
6
7      const strFromNum1 = String(num);
8      const strFromNum2 = String(77);
9      const strFromBool = String(true);
10     const numFromStr = Number("1900");
11     const boolFromNumber = Boolean(0);
12
13     console.log(strFromNum1);
14     console.log(strFromNum2);
15     console.log(strFromBool);
16     console.log(numFromStr);
17     console.log(boolFromNumber);
18
19 </script>

```

99

77

true

1900

false

Conversões para string

```

1
2 <meta charset="UTF-8">
3 <script>
4
5     let str1 = "USCS";
6     let str2 = String(str1);
7     console.log(`${typeof str1} : ${str1}`);           // -> string : USCS
8     console.log(`${typeof str2} : ${str2}`);           // -> string : USCS
9
10    let nr = 99;
11    let strNr = String(nr);
12    console.log(`${typeof nr} : ${nr}`);                // -> number : 99
13    console.log(`${typeof strNr} : ${strNr}`);           // -> string : 99
14
15    let bool1 = true;
16    let bool2 = String(bool1);
17    console.log(`${typeof bool1} : ${bool1}`);           // -> boolean : true
18    console.log(`${typeof bool2} : ${bool2}`);           // -> string : true
19
20 </script>
  
```

Conversões para string

```

1
2  <meta charset="UTF-8">
3  <script>
4
5      let bnr = 123n;
6      let strBnr = String(bnr);
7      console.log(`${typeof bnr} : ${bnr}`);           // -> bigint : 123
8      console.log(`${typeof strBnr} : ${strBnr}`);      // -> string : 123
9
10     let un = undefined;
11     let strUn = String(un);
12     console.log(`${typeof un} : ${un}`);              // -> undefined : undefined
13     console.log(`${typeof strUn} : ${strUn}`);         // -> string : undefined
14
15     let n = null;
16     let strN = String(n);
17     console.log(`${typeof n} : ${n}`);                // -> object : null
18     console.log(`${typeof strN} : ${strN}`);          // -> string : null
19
20 </script>

```


Conversões para number

```

2  <meta charset="UTF-8">
3  <script>
4      console.log(Number(99));           // -> 99
5      console.log(Number("11"));         // -> 11
6      console.log(Number("0x11"));       // -> 17
7      console.log(Number("0o11"));       // -> 9
8      console.log(Number("0b11"));       // -> 3
9      console.log(Number("12e3"));       // -> 12000
10     console.log(Number("Infinity"));   // -> Infinity
11     console.log(Number("text"));       // -> NaN
12
13     console.log(Number(14n));           // -> 14
14     console.log(Number(true));          // -> 1
15     console.log(Number(false));         // -> 0
16
17     console.log(Number(undefined));     // -> NaN
18     console.log(Number(null));          // -> 0
19 </script>
  
```

Conversão para boolean

- ✓ O valor **false** é sempre retornado para: 0, NaN, string vazio, undefined ou null;
- ✓ Qualquer outro valor sempre resultará em **true**.



Conversão para boolean

```

1
2  <meta charset="UTF-8">
3  <script>
4      console.log(Boolean(true));      // -> true
5      console.log(Boolean(false));     // -> false
6
7      console.log(Boolean(99));         // -> true
8      console.log(Boolean(0));          // -> false
9      console.log(Boolean(NaN));        // -> false
10
11     console.log(Boolean("USCS"));     // -> true
12     console.log(Boolean(""));         // -> false
13
14     console.log(Boolean(undefined));  // -> false
15     console.log(Boolean(null));       // -> false
16 </script>
  
```

Conversões implícitas

```

1
2  <meta charset="UTF-8">
3  <script>
4      const string1 = 99 + "1";
5      console.log(string1);           // -> 991
6      console.log(typeof string1);    // -> string
7
8      const string2 = 99 - "1";
9      console.log(string2);           // -> 98
10     console.log(typeof string2);     // -> number
11 </script>
  
```

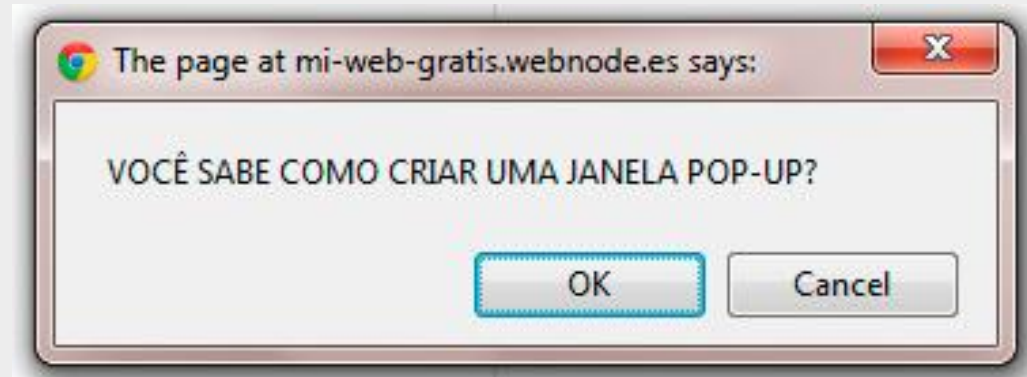
Interação com o usuário

- ✓ Programas **Javascript** escritos com **Node.js** para servidores não requerem interação com o usuário, sendo executados à nível de console, sem ambiente gráfico;
- ✓ No entanto, para programas **Javascript** client-side interface HTML é requerida juntamente com o uso do **DOM** (Document Object Model);



Interação com o usuário

- ✓ Programas Javascript também podem efetuar interação com o usuário por meio de caixas de diálogo (dialog boxes);
- ✓ Caixas de Diálogo fazem parte integrante dos browsers, sendo constituídas de janelas popup (ou modal) as quais quando exibidas aos usuários forçam a parada da interação enquanto a janela não for fechada.



Função alert()

```
1
2 <meta charset="UTF-8">
3 <script>
4     alert("Hello, World!")
5     window.alert("Hello, World novamente !!!");
6
7     alert(10 * 3);
8     alert(true);
9
10     alert("texto A", "texto B"); // somente texto A será exibido!
11 </script>
```


Função alert()

```

1
2 <meta charset="UTF-8">
3 <script>
4
5
6
7
8
9
10 alert("texto")
11 </script>

```

Essa página diz
Hello, World!

OK

Essa página diz
Hello, World novamente !!!

OK

Essa página diz
30

OK

Essa página diz
true

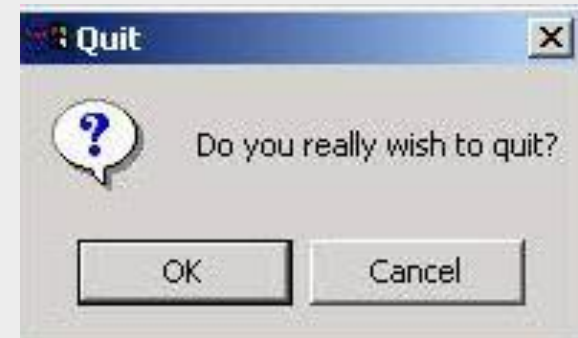
OK

Essa página diz
texto A

OK

Função confirm()

- ✓ É um método que aceita um parâmetro opcional (mensagem) e exibe dois botões (OK e Cancel);
- ✓ Dependendo do botão pressionado pelo usuário, o método retorna um valor booleano;
- ✓ **true** se for pressionado OK e **false** caso Cancel.



Função confirm()

```

1
2 <meta charset="UTF-8">
3 <script>
4
5     let remove = confirm("Remover todos os dados ?");
6     let mensagem = remove ? "Dados serão deletados!" : "Operação cancelada!";
7
8     console.log(mensagem);
9     alert(mensagem);
10 </script>
  
```

Essa página diz
Remover todos os dados ?

OK

Cancelar

Essa página diz
Dados serão deletados!

OK

Função prompt()

- ✓ Usada para exibir uma caixa de diálogo simples ao usuário para que este insira um valor ou uma resposta por meio de um campo de entrada.

```
1
2  <meta charset="UTF-8">
3  <script>
4
5      let resposta = prompt("Entre com seu time preferido:");
6      alert("Meu time preferido: " + resposta);
7
8  </script>
```

Função prompt()

```

1
2 <meta charset="UTF-8">
3 <script>
4
5     let resposta = prompt("Entre com seu time preferido:");
6     alert("Meu time preferido: " + resposta);
7
8 </script>
  
```

Essa página diz

Entre com seu time preferido:

Essa página diz

Meu time preferido: Juventos

Função prompt()

- ✓ Ao se executar a função `prompt()` uma caixa de diálogo é exibida ao usuário com a mensagem especificada dentro das aspas;
- ✓ Se o usuário clicar em "OK" e inserir um valor, esse valor será atribuído à variável resposta. Se o usuário clicar em "Cancelar" ou deixar o campo em branco, a variável resposta receberá o valor `null`;
- ✓ A função `prompt()` é frequentemente usada para obter entrada interativa do usuário.




Programação Sequencial

- ✓ Um programa sequencial é um programa que escrevemos a partir de três ações básicas e necessárias que um programa deve executar em um computador, que são:



Programação Sequencial

```

1
2 <meta charset="UTF-8">
3 <script>
4
5    let a = "Olá";
6   let b = " Mundo!";
7   let x = a + b;
8   alert(x);
9
10  let nome = prompt("Entre com o seu nome: ");
11  alert("Bom dia, " + nome + " !");
12
13 </script>
  
```

a Olá

Memória do Programa: os dados ficam na memória durante a execução do programa

Programação Sequencial

```

1
2 <meta charset="UTF-8">
3 <script>
4
5     let a = "Olá";
6     let b = " Mundo!";
7     let x = a + b;
8     alert(x);
9
10    let nome = prompt("Entre com o seu nome: ");
11    alert("Bom dia, " + nome + " !");
12
13 </script>
  
```

a Olá

b Mundo!

Memória do Programa: os dados ficam na memória durante a execução do programa

Programação Sequencial

```

1
2 <meta charset="UTF-8">
3 <script>
4
5     let a = "Olá";
6     let b = " Mundo!";
7     let x = a + b;
8     alert(x);
9
10    let nome = prompt("Entre com o seu nome: ");
11    alert("Bom dia, " + nome + " !");
12
13 </script>

```



a Olá

b Mundo!

x Olá Mundo!

Memória do
Programa: os dados
ficam na memória
durante a execução do
programa

Programação Sequencial

```

1
2 <meta charset="UTF-8">
3 <script>
4
5     let a = "Olá";
6     let b = " Mundo!";
7     let x = a + b;
8     alert(x);
9
10    let nome = prompt("Entre com o seu nome: ");
11    alert("Bom dia, " + nome + " !");
12
13 </script>

```

a Olá

b Mundo!

x Olá Mundo!


Memória do Programa: os dados ficam na memória durante a execução do programa

Essa página diz
Olá Mundo!

OK

Programação Sequencial

```

1
2 <meta charset="UTF-8">
3 <script>
4
5     let a = "Olá";
6     let b = " Mundo!";
7     let x = a + b;
8     alert(x);
9
10     let nome = prompt("Entre com o seu nome: ");
11    alert("Bom dia, " + nome + " !");
12
13 </script>

```

a Olá

b Mundo!

x Olá Mundo!

Ana Maria

nome

Memória do
Programa: os dados
ficam na memória
durante a execução do
programa

Essa página diz

Entre com o seu nome:

OK Cancelar

Programação Sequencial

```

1
2  <meta charset="UTF-8">
3  <script>
4
5      let a = "Olá";
6      let b = " Mundo!";
7      let x = a + b;
8      alert(x);
9
10     let nome = prompt("Entre com o seu nome: ");
11     alert("Bom dia, " + nome + " !");
12
13 </script>

```

a Olá

b Mundo!

x Olá Mundo!

Ana Maria

nome

Memória do
Programa: os dados
ficam na memória
durante a execução do
programa

Essa página diz

Bom dia, Ana Maria !

OK

Programação Sequencial

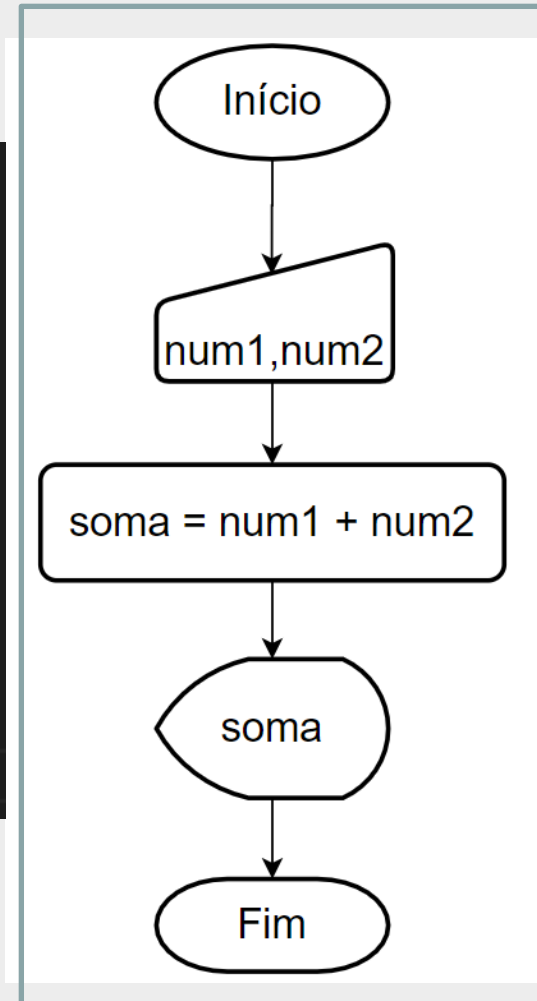
Programa que lê **2 números digitados** pelo usuário e exibe a **soma** deles.



Programação Sequencial

Programa que lê 2 números digitados pelo usuário e exibe a soma deles.

```
1
2 <meta charset="UTF-8">
3 <script>
4
5     let num1 = Number(prompt("Entre com o primeiro número:"));
6     let num2 = Number(prompt("Entre com o segundo número:"));
7
8     let soma = num1 + num2;
9
10    alert ("Soma = " + soma);
11
12 </script>
```



Fluxograma

Programação Sequencial

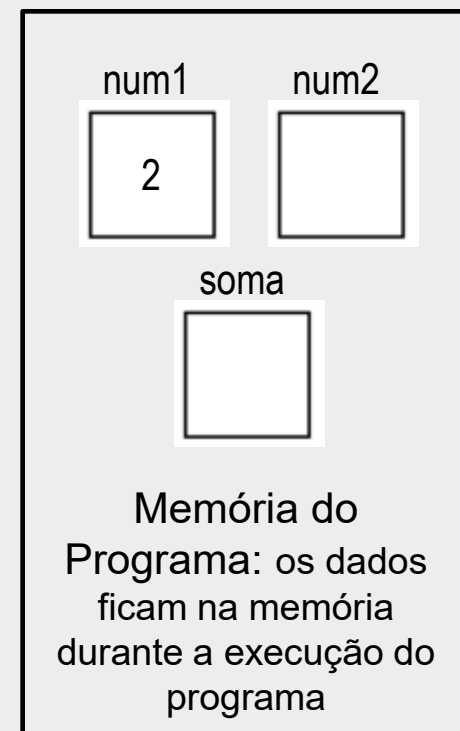
Programa que lê 2 números digitados pelo usuário e exibe a soma deles.

```
1
2 <meta charset="UTF-8">
3 <script>
4
5 let num1 = Number(prompt("Entre com o primeiro número:"));
6 let num2 = Number(prompt("Entre com o segundo número:"));
7
8 let soma = num1 + num2;
9
10 alert ("Soma = " + soma);
11
12 </script>
```

Essa página diz

Entre com o primeiro número:

OK Cancelar



Memória

Programação Sequencial

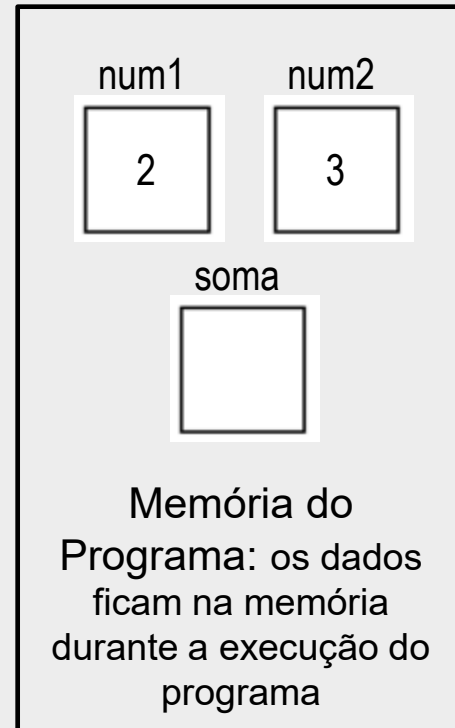
Programa que lê 2 números digitados pelo usuário e exibe a soma deles.

```
1
2 <meta charset="UTF-8">
3 <script>
4
5   let num1 = Number(prompt("Entre com o primeiro número:"));
6   let num2 = Number(prompt("Entre com o segundo número:"));
7
8   let soma = num1 + num2;
9
10  alert ("Soma = " + soma);
11
12 </script>
```

Essa página diz

Entre com o segundo número:

OK Cancelar

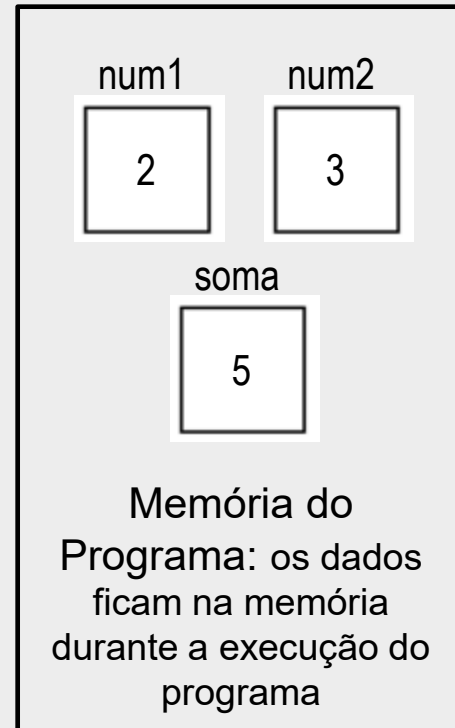


Memória

Programação Sequencial

Programa que lê 2 números digitados pelo usuário e exibe a soma deles.

```
1
2 <meta charset="UTF-8">
3 <script>
4
5     let num1 = Number(prompt("Entre com o primeiro número:"));
6     let num2 = Number(prompt("Entre com o segundo número:"));
7
8     let soma = num1 + num2;
9
10    alert ("Soma = " + soma);
11
12 </script>
```



Memória

Programação Sequencial

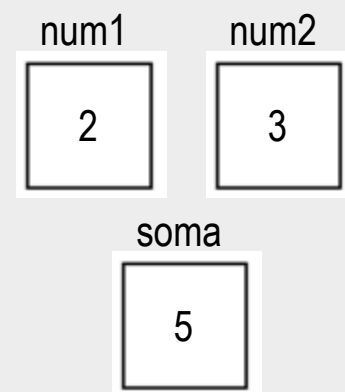
Programa que lê 2 números digitados pelo usuário e exibe a soma deles.

```
1
2 <meta charset="UTF-8">
3 <script>
4
5     let num1 = Number(prompt("Entre com o primeiro número:"));
6     let num2 = Number(prompt("Entre com o segundo número:"));
7
8     let soma = num1 + num2;
9
10    alert ("Soma = " + soma);
11
12 </script>
```

Essa página diz

Soma = 5

OK



Memória do
Programa: os dados
ficam na memória
durante a execução do
programa

Memória

Exercícios

1. Escrever um programa Javascript para calcular a área de um quadrado.
2. Escrever um programa Javascript para calcular a área de um trapézio.
3. Escrever um programa Javascript no qual o usuário digita o salário atual e o percentual do aumento salarial. O programa deverá exibir o novo salário do usuário.
4. Escrever um programa Javascript que leia um número entrado pelo usuário e exiba o dobro deste número.
5. Escrever um programa Javascript que leia dois números entrados pelo usuário e exiba o produto destes números.
6. Escrever um programa Javascript que leia o valor de um produto entrado pelo usuário e exiba o valor total do produto acrescido de 20%.

Exercícios

7. Escrever um programa Javascript que leia do usuário o tempo total de uma atividade expressa em horas. O programa deverá retornar o tempo correspondente à atividade expresso em minutos.
8. Escrever um programa Javascript que leia um número do usuário. O programa deverá exibir o valor anterior do número e o valor posterior. Por exemplo, se o usuário digitar 10, o programa deverá exibir 9 e 11.
9. Escrever um programa Javascript que leia do usuário o valor correspondente a um produto comprado por um cliente de uma loja. Considerando que o usuário irá pagar o produto em quatro parcelas sem juros, o programa deverá exibir o valor de cada parcela.
10. Escrever um programa Javascript que leia o preço normal de um produto de uma loja. O programa deverá informar o preço do produto com desconto de 10%.
11. Escrever um programa Javascript que leia as três notas de um aluno e exiba a média aritmética das notas entradas.