



# Sistema WEb

MATC82

QUARTAS e SEXTAS - 18:30 - 20:20

Carga Horária - Total: 68 horas

# Introdução a JavaScript

O Javascript é uma linguagem interpretada, isso significa que o código fonte é executado diretamente por um interpretador, que analisa o código linha por linha e executa as instruções em tempo real.

# Introdução a JavaScript

JavaScript é a linguagem de programação da Web.

A ampla maioria dos sites modernos usa JavaScript e todos os navegadores modernos – em computadores de mesa, consoles de jogos, tablets e smartphones.

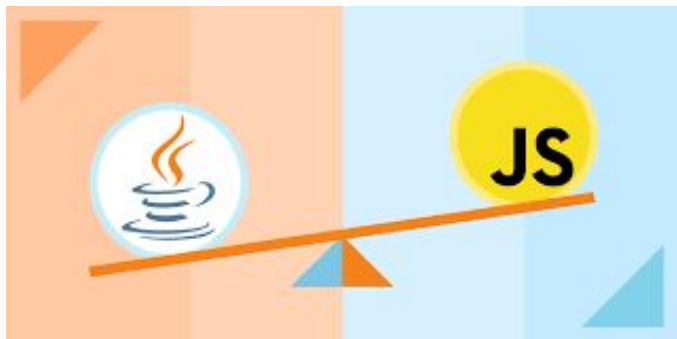
# O que é JavaScript?

- Linguagem de programação de alto nível.
- Usada para criar interação em páginas web.
- Executada no navegador (frontend) e também no servidor (Node.js).
- Popular por sua simplicidade e versatilidade.

# Introdução a JavaScript

O nome “JavaScript” é um pouco enganoso.

A não ser pela semelhança sintática superficial, JavaScript é completamente diferente da linguagem de programação Java.



Fonte: <https://learningdaily.dev/java-vs-javascript-whats-the-difference-7f6bde5fd7bd>

# Explorando JavaScript

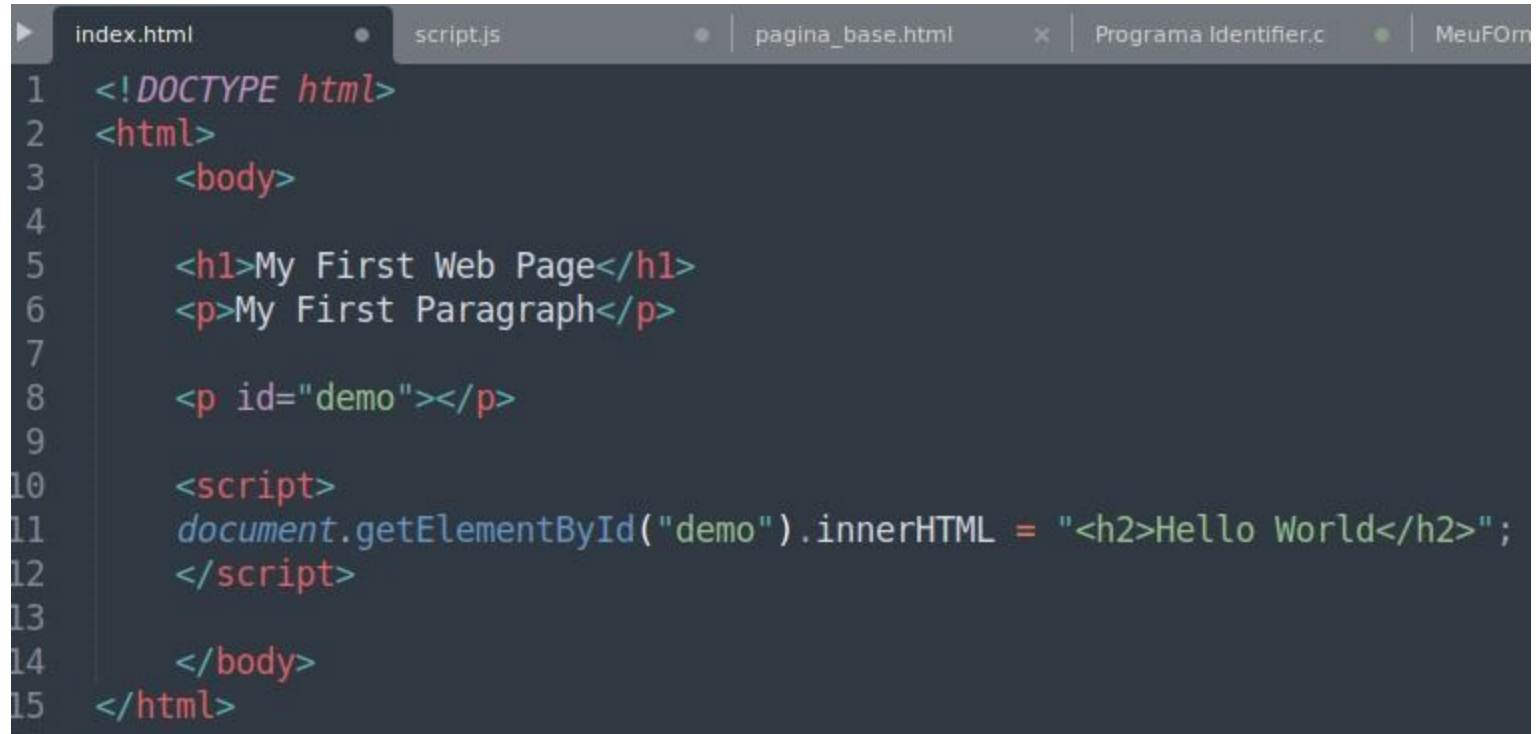
Você precisa de um interpretador de JavaScript. Felizmente, todo navegador Web contém um interpretador de JavaScript.



# Utilizando JavaScript:

- **Incorporando JavaScript:**
  - Inline: `<script>alert('Hello, World!');</script>`
  - Externo: `<script src="script.js"></script>`
- **Onde escrever? No início (head) ou no final (body).**

# Utilizando JavaScript:



The image shows a code editor with a dark background. At the top, there are several tabs: 'index.html' (active), 'script.js', 'pagina\_base.html', 'Programa Identifier.c', and 'MeuForm'. The 'index.html' tab is selected, and the code is as follows:

```
1  <!DOCTYPE html>
2  <html>
3      <body>
4
5      <h1>My First Web Page</h1>
6      <p>My First Paragraph</p>
7
8      <p id="demo"></p>
9
10     <script>
11         document.getElementById("demo").innerHTML = "<h2>Hello World</h2>";
12     </script>
13
14     </body>
15 </html>
```



# JavaScript básica

// Tudo que vem após barras normais duplas é um comentário em linguagem natural.

// As variáveis são declaradas com a palavra-chave var:

```
var x;
```

// Valores podem ser atribuídos às variáveis com o sinal =

```
x = 0;
```

ou /\* pode ser um parágrafo um um bloco do código\*/

# JavaScript básica

// JavaScript aceita vários tipos de valores

x = 1;

// Números.

x = 0.01;

// Apenas um tipo Number para inteiros e reais.

x = "hello world";

# JavaScript básica

```
//Strings de texto entre aspas.  
x = 'JavaScript';  
// Apóstrofos também delimitam strings.  
x = true;  
// Valores booleanos.  
x = false;  
// O outro valor booleano.
```

# Declaração de variáveis:

- `let nome = 'João';`
- `const idade = 25;`
- `var x;`

# Estrutura léxica

JavaScript é uma linguagem que diferencia letras maiúsculas de minúsculas.

A palavra-chave `while`, por exemplo, deve ser digitada como `"while"` e não como `"While"` ou `"WHILE"`.

# Palavras reservadas

JavaScript reserva vários identificadores como palavras-chave da própria linguagem. Você não pode usar essas palavras como identificadores em seus programas:

<code>break</code>	<code>delete</code>	<code>function</code>	<code>return</code>	<code>typeof</code>
<code>case</code>	<code>do</code>	<code>if</code>	<code>switch</code>	<code>var</code>
<code>catch</code>	<code>else</code>	<code>in</code>	<code>this</code>	<code>void</code>
<code>continue</code>	<code>false</code>	<code>instanceof</code>	<code>throw</code>	<code>while</code>
<code>debugger</code>	<code>finally</code>	<code>new</code>	<code>true</code>	<code>with</code>
<code>default</code>	<code>for</code>	<code>null</code>	<code>try</code>	

# Os nomes de variáveis não podem:

- Conter espaços.
- Começar por número.
- Conter caracteres especiais, como +, -, \*, /, %, (,), {, }, !, @, #.

# Pontos e vírgulas opcionais

- Em JavaScript, você normalmente pode omitir o ponto e vírgula entre duas instruções, caso essas instruções sejam escritas em linhas separadas.
- Muitos programadores JavaScript utilizam pontos e vírgulas para marcar explicitamente os finais de instruções, mesmo onde eles não são obrigatórios.



# Exemplos:

**a = 3;**

**b = 4;**

---

**var a**

**a**

**=**

**3**

**console.log(a)**

# Função simples:

```
function saudacao() {  
    console.log('Bem-vindo ao JavaScript!');  
}
```

# Estrutura léxica

```
<script>  
const nome = prompt("Qual é o seu nome?")  
alert("Olá " + nome)  
</script>
```

# Estrutura léxica – Cálculo do dobro de um número

```
<script>
```

```
const num = prompt("Número: ") // lê um dado de entrada
```

```
const dobro = num * 2
```

```
// calcula o dobro
```

```
alert("Dobro é: " + dobro) // exibe a resposta
```

```
</script>
```

# Atividade

Teste os códigos apresentados e crie: uma página de adivinhação.

# Operadores

- Tipos de operadores
  - Operadores Aritméticos
  - Operadores de Atribuição
  - Operadores de comparação
  - Operadores Lógicos

# Operadores Aritméticos

- Usados para realizar operações aritméticas em números

Operador	Descrição
+	Adição
-	Subtração
*	Multiplificação
**	Exponenciação
/	Divisão
%	Módulo
++	Incremento
--	Decremento

# Operadores de Atribuição

- Atribuem valores a variáveis JavaScript
  - ( += ) atribuição de adição adiciona um valor a uma variável

Operador	Descrição
=	Atribuição
+=	Adição e Atribuição
-=	Subtração e Atribuição
*=	Multiplicação
/=	Divisão e Atribuição
%=	Módulo e Atribuição
**=	Exponenciação e Atribuição



# Operadores de Comparação

- Usados em declarações lógicas para determinar igualdade ou diferença entre variáveis ou valores
  - usados para testar true ou false.

Operador	Descrição
==	igual a
===	valor igual e tipo igual
!=	diferente
!==	valor diferente ou tipo diferente
>	maior que
<	menor que
>=	maior ou igual a
<=	menor ou igual a
?	operador ternário

# Operadores Lógicos

- Usados para determinar a lógica entre variáveis ou valores.
  - usados para testar true ou false.

Operador	Descrição
&&	And
	Or
!	Not

# Tipos de dados JavaScript

- Variável JavaScript pode conter qualquer tipo de dado.

```
1 // Números:
2 let idade = 16;
3 let peso = 7.5;
4
5 // Strings:
6 let cor = "verde";
7 let nome = "Laise";
8
9 // Booleanos
10 let x = true;
11 let y = false;
12
13 // Objeto:
14 const aluno = {primeiroNome:"Laise", segundoNome:"Cavalcante"};
15
16 // Objeto array:
17 const carros = ["Creta", "HB20", "L200"];
18
19 // Objeto Date:
20 const dataEntrada = new Date("2022-03-25");
```

# Estrutura condicional

- Usadas para executar diferentes ações com base em diferentes condições

if	se uma condição especificada for verdadeira
else	se a mesma condição for falsa
else if	se a primeira condição for falsa
switch	especificar muitos blocos alternativos

# Declaração if

- Sintaxe

```
1  if (condição) {  
2    // bloco de código a ser executado se a condição for verdadeira  
3  }
```

- Exemplo

```
1  if (idade < 18) {  
2    mensagem = "Não pode entrar";  
3  }
```

# Declaração else

- Sintaxe

```
1  if (idade < 18) {  
2    mensagem = "Não pode entrar";  
3  
4  } else {  
5    mensagem = "Pode entrar";  
6  
7  }
```

# Declaração else if

- Sintaxe

```
1  if (hora < 10) {  
2      mensagem = "Bom dia";  
3  
4  } else if (hora < 18) {  
5      mensagem = "Boa tarde";  
6  
7  } else {  
8      mensagem = "Boa noite";  
9  }
```

# Declaração Switch

- Sintaxe

```
1  switch (new Date().getDay()) {  
2      case 6:  
3          text = "Hoje é sábado";  
4          break;  
5      case 0:  
6          text = "Hoje é domingo";  
7          break;  
8      default:  
9          text = "Ansiosos pelo fim de semana";  
10 }
```



# Estrutura de repetição

- Úteis se você quiser executar o mesmo código repetidamente.
  - podem executar um bloco de código diversas vezes.
  - geralmente esse é o caso ao trabalhar com array.

# Estrutura de repetição For

- Sintaxe

```
1  for (expressão 1; expressão 2; expressão 3) {  
2  // bloco de código a ser executado  
3  }
```

# Estrutura de repetição For

- Exemplo

```
1  for (let i = 0; i < 5; i++) {  
2    text += "número " + i + "<br>";  
3  }
```

# Estrutura de repetição While

- Percorre um bloco de código enquanto uma condição especificada for verdadeira.
  - Sintaxe

```
1 while (condição) {  
2 // bloco de código a ser executado  
3 }
```

# Estrutura de repetição While

- Exemplo

```
1  while (i < 10) {  
2      text += "Número: " + i;  
3      i++;  
4  
5  }
```

# Estrutura de repetição Do While

- Sintaxe

```
1  do {  
2  // bloco de código a ser executado  
3  }  
4  while (condição);  
5
```

# Estrutura de repetição Do While

- Exemplo

```
1  do {  
2      text += "Número: " + i;  
3      i++;  
4  }  
5  while (i < 10);
```

# Array JavaScript

- Um Array é uma variável especial, que pode conter mais de um valor por vez.
  - armazenar muitos valores com um único nome(identificador).
  - acessar os valores consultando um número de índice.



# Array JavaScript

- Sintaxe

```
1  var array_name = [item1, item2, ...]  
2
```

- Exemplo

```
1  var carros = [ "Saab", "Volvo", "BMW" ]  
2  
3  var carros = new Array("Saab", "Volvo", "BMW")  
4  
5  var pessoa = [ "Carlos", "Silva", 46 ];  
6
```

# Funções JavaScript

- Uma função JavaScript é um bloco de código projetado para executar uma tarefa específica.
- Uma função JavaScript é executada quando "algo" a invoca (a chama).
  - Exemplo

```
1 // Função para calcular o produto de p1 e p2
2 function nomeFuncao(p1, p2) {
3     return p1 * p2;
4
5 }
```

# Funções JavaScript

- Sintaxe

```
1  function nomeFuncao(parametro1, parametro12, parametro13) {  
2      // código a ser executado  
3  
4  }
```

- Definida com a palavra-chave `function`, seguida de um nome , seguido de parênteses `()` .
- Os parênteses podem incluir nomes de parâmetros separados por vírgulas:
  - `( parâmetro1, parâmetro2, ... )`

# Invocação de função

- O código dentro da função será executado quando "algo" invocar (chamar) a função:
  - Quando um evento ocorre
    - i. clica em um botão
  - Quando é invocado (chamado) a partir do código JavaScript
  - Automaticamente (auto invocado)

# Função com Retorno

- Exemplo

```
1 // A função é chamada, o valor de retorno terminará em x
2 let x = multiplicacao(4, 3);
3
4 function multiplicacao(a, b) {
5     // A função retorna o produto de a e b
6     return a * b;
7
8 }
```

# Eventos JavaScript

- O que são eventos em JavaScript
- Associar eventos ao HTML
- Exemplos práticos
- Principais eventos
- Utilizando eventos em Forms

# Eventos em JavaScript

Um evento é a ocorrência de uma ação, geralmente produzida por um usuário, em uma página.

Clicar em um botão, selecionar um item, sair de um campo, pressionar uma tecla, passar o mouse sobre uma imagem.

# Tipos de eventos JavaScript

- Eventos de mouse (onclick, ondblclick, onmouseover)
- Eventos de teclado (keypress, keydown, keyup)
- Eventos de formulário (change, focus, blur)



## Associar eventos aos elementos HTML

- Utilizar o atributo on diretamente na tag dentro do arquivo HTML.
- Usar métodos do próprio JavaScript de adição e remoção de eventos como o **addEventListener** o **removeEventListener**.

# Estrutura dos eventos

- Estrutura básica que envolve: Event Target, Event Listener e Event Handler.

# Event Target

- O Event Target é o elemento onde o evento vai acontecer.
  - Exemplo: a tag `<button>` no HTML, uma imagem, um link ou até mesmo a própria janela do navegador.
  - Exemplo

```
1 <button id="meu-botao"> Clique aqui </button>
```

# Event Listener

- É a função que aguarda até que um evento ocorra em um elemento.
  - adicionado através do método **addEventListener**
  - fica alerta para qualquer evento que aconteça
  - exemplo, podemos adicionar um Event Listener no botão para que ao clicar nele

```
1 let button = document.getElementById('meu-botao');  
2 button.addEventListener('click', () => {  
3     alert('Botão clicado!');  
4 });
```

# Event Handler

- O Event Handler é a ação, representada por uma função ou código, que ocorre em resposta ao evento.
  - Event Handler também pode ser uma função definida separadamente.

```
1  function exibirMensagem() {  
2    alert('Botão clicado!');  
3  }  
4  
5  let button = document.getElementById('meu-botao');  
6  button.addEventListener('click', exibirMensagem);
```

# Evento de click

- Evento acontece quando o usuário clica em um elemento da página

```
1  <!DOCTYPE html>
2  <html lang="pt-Br">
3      <head>
4          <meta charset="UTF-8">
5          <meta name="viewport" content="width=device-width, initial-scale=1.0">
6          <title>Evento de clique</title>
7      </head>
8
9      <body>
10         <button id="meu-botao">Clique aqui</button>
11         <script src="evento_click.js"></script>
12     </body>
13
14 </html>
15
```

# Evento de click

- .JS

```
1  var botao = document.querySelector("#meu-botao")
2  botao.addEventListener('click', () => {
3    alert('Botão clicado!');
4  })
```

# Evento de mouseover

- Evento acontece quando o cursor do mouse passa por cima de um elemento da página

```
1 <!DOCTYPE html>
2 <html lang="pt-Br">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Evento de mouseover</title>
7   </head>
8
9   <body>
10    <ul id="lista">
11      <li>Item 1</li>
12      <li>Item 2</li>
13      <li>Item 3</li>
14    </ul>
15    <script src="evento_mouseover.js"></script>
16  </body>
17
18 </html>
```



# Evento de mouseover

- .JS

```
1  var lista = document.querySelector("#lista");
2
3  lista.addEventListener("mouseover", (event)=>{
4      event.target.style.color = "orange";
5      setTimeout(() => {
6          event.target.style.color = "";
7      }, 500);
8  })
```

# Evento de mouseover

- Evento acontece quando o cursor do mouse passa por cima de um elemento da página

```
1 <!DOCTYPE html>
2 <html lang="pt-Br">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Evento de mouseover</title>
7   </head>
8
9   <body>
10    <ul id="lista">
11      <li>Item 1</li>
12      <li>Item 2</li>
13      <li>Item 3</li>
14    </ul>
15    <script src="evento_mouseover.js"></script>
16  </body>
17
18 </html>
```

# Evento de mouseover

- .JS

```
1  var lista = document.querySelector("#lista");
2
3  lista.addEventListener("mouseover", (event)=>{
4      event.target.style.color = "orange";
5      setTimeout(() => {
6          event.target.style.color = "";
7      }, 500);
8  })
```

# Evento de submit

- Disparado quando um formulário é submetido.
  - Por padrão, ao submeter um formulário, a página é automaticamente recarregada.
  - O evento submit é disparado no próprio elemento `<form>`
  - O evento `SubmitEvent` enviado para indicar a ação possui uma propriedade `submitter`, que é o botão que realizou o pedido de submit.
  - Disparado quando a pessoa usuária clica em um botão (`<button>` ou `<input type="submit">`) ou pressiona “ENTER”

# Evento de submit

- Exemplo

```
1  <!DOCTYPE html>
2  <html lang="pt-Br">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Evento de submit</title>
7  </head>
8  <body>
9      <form id="meu-formulario">
10         <input id="email" type="text" placeholder="E-mail">
11         <input id="senha" type="text" placeholder="Senha">
12         <button type="submit">Submeter o formulário</button>
13     </form>
14     <script src="evento_submit.js"></script>
15 </body>
16 </html>
```

# Evento de submit

- .JS

```
1  var formulario = document.querySelector('#meu-formulario')
2
3  formulario.addEventListener("submit", (event) =>{
4      event.preventDefault();
5      alert('Formulário enviado!');
6  })
7
```

# Evento change

- Disparado em tags como <input>, <select> e <textarea>

```
1 <!DOCTYPE html>
2 <html lang="pt-Br">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Evento de change</title>
7 </head>
8 <body>
9   <label>
10     Escolha seu sabor de sorvete favorito!
11     <select id="sorvete" name="sorvete">
12       <option value="">Selecione um sabor:</option>
13       <option value="chocolate">Chocolate</option>
14       <option value="morango">Morango</option>
15       <option value="creme">Creme</option>
16     </select>
17   </label>
18   <div id="resultado"></div>
19   <script src="main.js"></script>
20 </body>
21 </html>
```

# Evento change

- .JS

```
1  var elementoSelecioneado = document.querySelector("#sorvete");
2  var resultado = document.querySelector("#resultado");
3
4  elementoSelecioneado.addEventListener("change", (event) => {
5    resultado.textContent = `Você escolheu ${event.target.value}`;
6  });
```



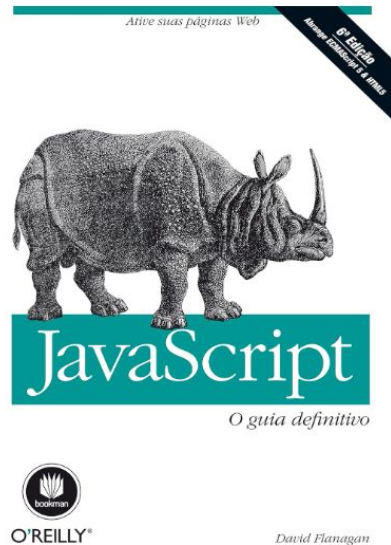
# Atividade

- Pesquisar e testar sintaxes JavaScript
  - Propriedade e métodos de um Array
  - Acesso aos elementos de um Array
  - Arrays como objetos
  - Conjuntos set
  - Criar funções e chamadas de teste.

# Referências:



HTML: A Linguagem de M



**Obrigada !**