

# JAVASCRIPT

Professor Paulo Henrique de Oliveira



JS

Ultima atualização: 14/06/2021 13:46

# Conteúdo Programático

- Comandos básicos
- Variáveis
- Tipos de dados
- Operadores lógicos
- Operadores relacionais
- Estruturas condicionais
- Estruturas de repetição
- Funções





# Olá, Mundo!

Eu sou o Professor Paulo Henrique

Ou se preferir, **PH**

Encontre-me nas redes sociais



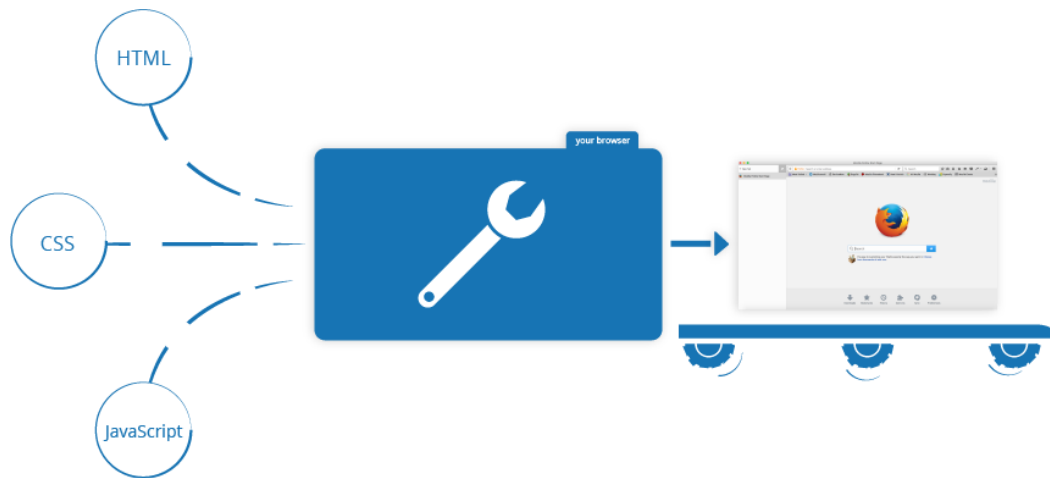
**@olamundoph**

# O que é javascript?

- JavaScript é uma linguagem de programação que permite a criação de conteúdo que se atualiza dinamicamente, controlar multimídias, imagens animadas, etc.
- É a terceira camada do bolo das tecnologias padrões da web, depois da HTML e CSS.

# HTML + CSS + JS

- HTML, CSS e JavaScript trabalham juntas. Quando uma página web é carregada no navegador, o código (HTML, CSS e JavaScript) é executado dentro de um ambiente de execução (a guia do navegador). Isso é como uma fábrica que pega a matéria prima (o código) e transforma em um produto (a página web).



## Como adicionar javascript no código

- O JavaScript é inserido na página de uma maneira similar ao CSS. Enquanto o CSS tem o elemento `<link>` para aplicar folhas de estilo externas e o elemento `<style>` para aplicar folhas de estilo internas, o JavaScript só precisa do elemento `<script>`.

# Como adicionar javascript no código

- Interno:

```
1  <script>
2
3      // O código JavaScript fica aqui
4
5  </script>
```

# Como adicionar javascript no código

- Externo:

```
1  <!DOCTYPE html>
2  <html lang="pt-br">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Document</title>
8  </head>
9  <body>
10
11      <script src="arquivo.js" defer></script>
12  </body>
13  </html>
```

O atributo *defer*, informa ao *browser* para continuar renderizando o conteúdo HTML uma vez que a tag `<script>` foi atingida.



# Comentários no código

- Há duas maneiras de comentário no código javascript. Pode ser em uma única linha ou em múltiplas linhas.
- Dessa forma o desenvolvedor pode criar comentários que ajudem outros desenvolvedores a compreender melhor o código.

```
1 // Comentário em uma linha
2
3 /*
4  Comentário
5  em
6  várias
7  linhas
8  */
```

# Primeiros passos – janela alert.

- O javascript permite criar de maneira bem simples, janelas que abrem no navegador assim que a página é executada.
- Código da janela ALERT:

```
<script>  
|   alert('Olá, MUNDO!')  
</script>
```

Resultado:

Javascript/Atividades/001.html

Essa página diz

Olá, MUNDO!

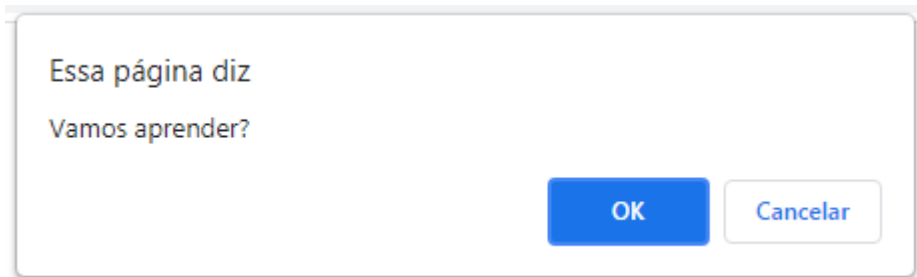
OK

# Primeiros passos – janela CONFIRM.

- Código da janela CONFIRM:

```
<script>  
|   confirm('Vamos aprender?')  
</script>
```

Resultado:



# Primeiros passos – janela PROMPT.

- Código da janela PROMPT:

```
<script>  
|   prompt('Qual o seu nome?')  
</script>
```

Resultado:

/Javascript/Atividades/001.html

Essa página diz

Qual o seu nome?

OK

Cancelar

# Variáveis

- Uma variável é um container para um valor, como um número que podemos usar em uma operação de adição, ou uma sequência de texto que possamos usar como parte de uma oração. Mas uma coisa especial a respeito das variáveis é que seu conteúdo pode mudar.
- Exemplo, em um sistema onde são cadastrados alunos. É certa que em todos os cadastros será registrado o nome do aluno, então **nome** é um dado fixo, porém, em cada registro esse nome é diferente do outro. É uma informação variável.
- A variável é chamada com o comando **var**.

# Identificadores das variáveis

Quando se cria uma variável é necessário colocar um nome nela, que chamamos de identificador. É importante saber que existem algumas regrinhas na hora de escolher esse nome:

- Podem começar com letra, \$ ou \_
- Não podem começar com números
- É possível usar letras e números
- É possível usar acentos e símbolos
- Não podem conter espaços
- Não podem ser palavras reservadas (ex: var e function)

# Dicas para identificadores (boas práticas)

- Maiúsculas e minúsculas fazem diferença, preste sempre muita atenção
- Escolha nomes que façam sentido com a informação dela
- É boa prática ao ter duas palavras, que a segunda comece em maiúscula, ex: `var meuNome`
- Normalmente não se usa acentos nos identificadores por já ser um hábito comum em outras linguagens que não permitem



# Tipos de dados - Números

- É possível armazenar números em variáveis, tanto números inteiros, como por exemplo 30 (também chamados de *integers*) como números decimais, por exemplo 2.456 (também chamados de *floats* ou *floating point numbers*).
- Não precisa declarar tipos de variáveis no JavaScript, diferentemente de outras linguagens de programação. Quando você atribui a uma variável o valor em número, você não inclui as aspas.

```
1 var minhaIdade = 29;
```



# Tipos de dados - Strings

- *Strings* são sequências de texto. Quando você dá a uma variável um valor em texto (*string*), você precisa envolver o texto em aspas simples, duplas ou crases; caso contrário, o JavaScript vai tentar interpretá-lo como sendo outro nome de variável.

```
1 var fraseDoDia = 'Vou aprender javascript';
```

# Tipos de dados - Booleans

- *Booleans* são valores verdadeiro/falso (*true/false*) – eles podem ter dois valores, *true* (verdadeiro) ou *false* (falso).
- São geralmente usados para verificar uma condição, que em seguida o código é executado apropriadamente.

```
1  var estouAcordado = true;  
2  
3  var comparar = 6 < 3;
```

# Tipos de dados - Arrays

- Um array é um único objeto que contém valores múltiplos inseridos entre colchetes e separados por vírgulas.
- Uma vez que esses arrays estejam definidos, é possível acessar cada um de seus valores através de sua localização dentro do array, indicando sua posição (a primeira posição é sempre a 0).

```
1  var nomeArray = ['Paulo', 'Bob', 'Jim'];
2  var idadeArray = [20,15,29];
3
4  nomeArray[0]; // deve retornar 'Paulo'
5  idadeArray[2]; // deve retornar 29
```

# Objetos

- Em programação, um objeto é uma estrutura de código que representa um objeto da vida real. Pode ter um simples objeto que representa um estacionamento de carro contendo informações sobre sua largura e comprimento, ou ter um objeto que representa uma pessoa, e contém dados sobre seu nome, altura, peso, que idioma ela fala, etc.

```
1  var cachorro = { nome : 'Code', raca : 'Shitzu' };  
2  
3  cachorro.nome // acessando a informação que tem em nome da variável cachorro
```

# Concatenação

- Concatenar significa "colocar junto". Para colocar strings juntas em JavaScript, usa-se o operador (+), o mesmo usado para adicionar números, mas neste contexto é algo diferente.

```
12      var saudacao = "Olá, "  
13      var pergunta = 'tudo bem?'  
14      var frase = saudacao + pergunta  
15      alert(frase)
```

# Concatenação

- Vamos ver a concatenação em um contexto dinâmico:

```
11     <script>
12         var nome = prompt('Qual é o seu nome?');
13         alert('Olá ' + nome + ', prazer em conhecê-lo!');
14     </script>
15 </body>
16 </html>
```

Resultado:

Essa página diz

Qual é o seu nome?

OK Cancelar

Essa página diz

Olá PH, prazer em conhecê-lo!

OK

# Converter de string para número - **Number**

- Nesse primeiro exemplo, o alert retorna os números concatenados(unidos) e não somados, pois n1 e n2 são variáveis do tipo strings (por estarem em aspas):

```
<script>  
  var n1= "5"  
  var n2= "2"  
  alert(n1 + n2)  
</script>
```

Essa página diz  
52

OK

- Nesse outro exemplo, as strings são convertidas para números através de Number(var) e agora os valores são somados e não concatenados, já que são valores numéricos.

```
<script>  
  var n1= "5"  
  var n2= "2"  
  alert(Number(n1) + Number(n2))  
</script>
```

Essa página diz  
7

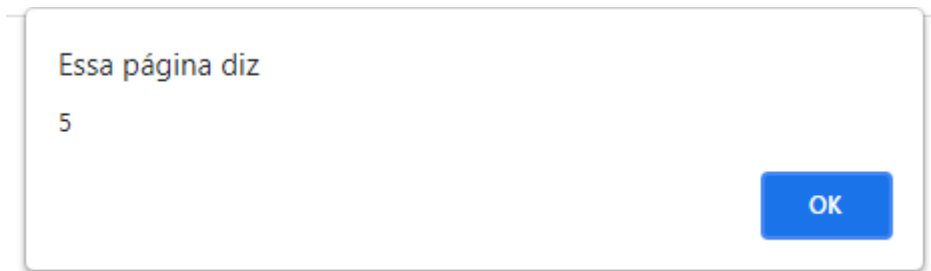
OK

# Converter para real e inteiro- **parseInt** e **parseFloat**

- Para converter um número real para inteiro, basta usar `Number.parseInt`

```
<script>  
  var meuNumero = 5.5  
  alert(Number.parseInt(meuNumero))  
</script>
```

Resultado:



- Para converter algum tipo para real, basta usar `Number.parseFloat`



# Converter para String

- Para converter um dado para String é possível de duas maneiras, utilizando o `String(var)` ou o `var.toString`

```
var meuNumero = 86
// Número para String
String(meuNumero)
// OU
meuNumero.toString
```

# Template Strings

- Template Strings é uma alternativa a concatenação de strings e variáveis para quem preferir não ter que usar o operador (+) toda vez que for trocar entre string e variável.
- Usa-se crase em vez de aspas e antes de para cada variável usa-se \${var}
- Nos dois casos o resultado é o mesmo.

```
var nome = "PH"
var idade = 29
var trabalho = "Professor"

// concatenando strings e variaveis
var fraseConcatenada = 'O ' + nome + ' tem ' + idade + ' anos de idade e é ' + trabalho
alert(fraseConcatenada)

// utilizando TEMPLATE STRING para o mesmo resultado, em vez de aspas usa-se crase
var fraseTemplate = `O ${nome} tem ${idade} anos de idade e é ${trabalho}`
alert(fraseTemplate)
```

Essa página diz

O PH tem 29 anos de idade e é Professor

OK

# Identificar o tipo de dado - typeof

- Utiliza-se **typeof** para retornar o seu tipo de dado

```
var nome = "PH"  
var tipo1 = typeof nome  
alert(tipo1)  
  
var idade = 29  
var tipo2 = typeof idade  
alert(tipo2)
```

Essa página diz  
string

OK

Essa página diz  
number

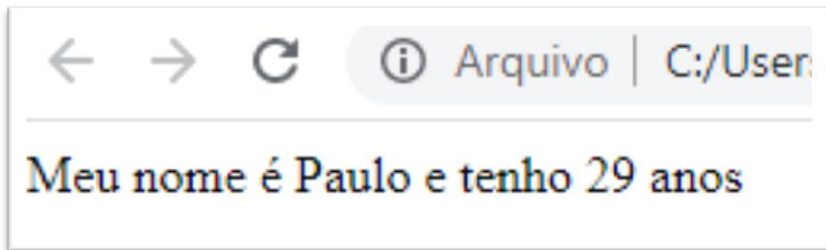
OK

# Enviar/Imprimir no HTML – document.write

- Para emitir alguma mensagem ao html com o javascript, é utilizado document.write("mensagem")

```
var nome = "Paulo"  
var idade = 29  
document.write("Meu nome é " + nome + " e tenho " + idade + " anos")
```

Resultado:



# Formatação de Strings

- É possível verificar a quantidade de caracteres de uma variável através de `.length`
- Converter todas letras em maiúsculas com `.toUpperCase()`
- Converter todas letras em minúsculas com `.toLowerCase()`

```
var nome = "Paulo"  
var quant = nome.length  
var maiusc = nome.toUpperCase();  
var minusc = nome.toLowerCase();  
document.write("Meu nome é tem " + quant + " letras. Em maiusculo fica: " + maiusc + ". Em minusculo fica: " + minusc)
```

Meu nome é tem 5 letras. Em maiusculo fica: PAULO. Em minusculo fica: paulo

# Formatação de Números

- É possível forçar um número a ter mais casas decimais com `.toFixed()` e o numero de casas dentro dos parênteses
- E para substituir ponto por vírgula, basta usar `.replace('.', ',')`

```
var numero = 163.8  
var num1 = numero.toFixed(2)  
var num2 = numero.toFixed(2).replace('.', ',')  
document.write("Com duas casas decimais: " + num1 + " e com vírgula: " + num2)
```

---

Com duas casas decimais: 163.80 e com vírgula: 163,80

# Operadores aritméticos

	Descrição	Exemplo		Resultado	
+	Soma dos valores	var = 5 + 5		var = 10	
-	Subtração dos valores	var = 15 - 5		var = 10	
*	Multiplicação dos valores	var = 2 * 5		var = 10	
/	Divisão dos valores	var = 10 / 5		var = 2	
%	Resto de uma divisão	var = 12 % 5		var = 2	
++	Adiciona 1 a um valor	var = 2 var++	var = 2 ++var	var = 2 e em seguida var = 3	var = 3
--	Subtrai 1 a um valor	var = 2 var--	var = 2 --var	var = 2 e em seguida var = 1	var = 1
**	Potência de um valor	var = 3 ** 2		var = 9	
-var	Negativa de uma variável	var = 2 -var		var = -2	

# Operadores aritméticos

```
var n1 = 5 + 5
var n2 = 15 - 5
var n3 = 2 * 5
var n4 = 10 / 5
var n5 = 12 % 5
var n6 = 2 // n6++
var n7 = 2 // ++n7
var n8 = 2 // n8--
var n9 = 2 // --n9
var n10 = 3 ** 2
var n11 = 2 // -n11
document.write("n1 = " + n1 + " | " + // para facilitar visualizar fiz quebra de linha no código usando +
    "n2 = " + n2 + " | " + // o + tem que ficar no final da linha, antes de fazer a quebra
    "n3 = " + n3 + " | " + // a quebra de linha está apenas no código e não no resultado HTML
    "n4 = " + n4 + " | " +
    "n5 = " + n5 + " | " +
    "n6 = " + n6++ + " | " +
    "n7 = " + ++n7 + " | " +
    "n8 = " + n8-- + " | " +
    "n9 = " + --n9 + " | " +
    "n10 = " + n10+ " | " +
    "n11 = " + -n11)
```

Resultado:

n1 = 10 | n2 = 10 | n3 = 10 | n4 = 2 | n5 = 2 | n6 = 2 | n7 = 3 | n8 = 2 | n9 = 1 | n10 = 9 | n11 = -2



# Operadores de auto atribuição

- Um operador de atribuição atribui um valor ao operando à sua esquerda baseado no valor do operando à direita. O operador de atribuição básico é o igual (=), que atribui o valor do operando à direita ao operando à esquerda. Nos exemplos a **var x** inicialmente vale **5**.

**var x = 5**

Auto atribuição	Encurtado	Equivalente	Resultado após a auto atribuição
Atribuição de adição	x += 2	x = x + 2	x = 7
Atribuição de subtração	x -= 2	x = x - 2	x = 3
Atribuição de multiplicação	x *= 2	x = x * 2	x = 10
Atribuição de divisão	x /= 2	x = x / 2	x = 2.5
Atribuição de resto	x %= 2	x = x % 2	x = 1
Atribuição exponencial	x **= 2	x = x ** 2	x = 25

# Operadores de auto atribuição

```
var n1 = 5
n1 += 2
document.write("n1 = " + n1 + "<br>" ) // <br> no final para fazer quebra de linha no HTML
var n2 = 5 // poderia ser utilizado apenas um document.write e também fazer quebras de linha no código
n2 -= 2 // porém aqui está sendo mostrado mais uma alternativa para aprender
document.write("n2 = " + n2 + "<br>" )
var n3 = 5
n3 *= 2
document.write("n3 = " + n3 + "<br>" )
var n4 = 5
n4 /= 2
document.write("n4 = " + n4 + "<br>" )
var n5 = 5
n5 %= 2
document.write("n5 = " + n5 + "<br>" )
var n6 = 5
n6 **= 2
document.write("n6 = " + n6 + "<br>" )
```

Resultado:

n1 = 7  
n2 = 3  
n3 = 10  
n4 = 2.5  
n5 = 1  
n6 = 25

# Ordem de precedência dos operadores aritméticos

- Ao usar os operadores aritméticos, existe uma ordem de precedência, caso apareça mais de uma operação na mesma linha. Então vai ser calculada a precedência superior e em seguida as demais, assim como em uma expressão comum da matemática.

<b>()</b>	Primordialmente calcula o que está dentro de parênteses
<b>**</b>	A segunda ordem é a potência
<b>* / %</b>	A terceira ordem é a multiplicação, divisão e módulo
<b>+ -</b>	A quarta ordem é a adição e a subtração

- Caso tenha mais de uma operação da mesma ordem, é calculado a que vier primeiro da esquerda para a direita. Exemplo: Em **var = 4 \* 5 / 4**, a multiplicação é feita antes da divisão.

# Operadores Relacionais

- Operadores relacionais são operadores de comparação.

**var x = 10**

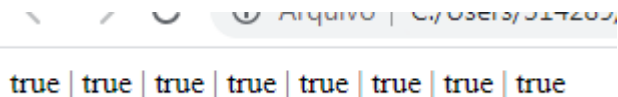
Operador	Descrição	Exemplos que retornam verdadeiro
<b>Igual (==)</b>	Retorna verdadeiro caso os operandos sejam iguais.	x == '10'
<b>Não igual (!=)</b>	Retorna verdadeiro caso os operandos não sejam iguais.	x != 8
<b>Estritamente igual (===)</b>	Retorna verdadeiro caso os operandos sejam iguais e do mesmo tipo.	x === 10
<b>Estritamente não igual (!==)</b>	Retorna verdadeiro caso os operandos não sejam iguais e/ou não sejam do mesmo tipo.	x !== "10"
<b>Maior que (&gt;)</b>	Retorna verdadeiro caso o operando da esquerda seja maior que o da direita.	x > 2
<b>Maior que ou igual (&gt;=)</b>	Retorna verdadeiro caso o operando da esquerda seja maior ou igual ao da direita.	x >= 10
<b>Menor que (&lt;)</b>	Retorna verdadeiro caso o operando da esquerda seja menor que o da direita.	x < 12
<b>Menor que ou igual (&lt;=)</b>	Retorna verdadeiro caso o operando da esquerda seja menor ou igual ao da direita.	x <= 10

# Operadores Relacionais

- Exemplo com valores, onde retornam **true**:

```
var x = 10
var r1 = (x == '10')
var r2 = (x != 8)
var r3 = (x === 10)
var r4 = (x !== '10')
var r5 = (x > 2)
var r6 = (x >= 10)
var r7 = (x < 12)
var r8 = (x <= 10)
document.write(r1 + " | " + r2 + " | " + r3 + " | " + r4 + " | " + r5 + " | " + r6 + " | " + r7 + " | " + r8)
```

- Resultado:



true | true | true | true | true | true | true | true

# Operadores Relacionais

- Exemplo com outros valores, onde retornam **false**:

```
var x = 10
var r1 = (x == 9)
var r2 = (x != 10)
var r3 = (x === "10")
var r4 = (x !== 10)
var r5 = (x > 11)
var r6 = (x >= 15)
var r7 = (x < 8)
var r8 = (x <= 5)
document.write(r1 + " | " + r2 + " | " + r3 + " | " + r4 + " | " + r5 + " | " + r6 + " | " + r7 + " | " + r8)
```

- Resultado:

false | false | false | false | false | false | false | false

# Operadores Lógicos

Operador	Leia-se	Utilização	Descrição
<b>&amp;&amp;</b>	<b>Conjunção</b> <b>E</b>	var1 && var2  true && true = true true && false = false false && false = false	(E lógico) - Retorna verdadeiro caso ambos operandos sejam verdadeiros; caso contrário, retorna falso.
<b>  </b>	<b>Disjunção</b> <b>OU</b>	var1    var2  true    true = true true    false = true false    false = false	(OU lógico) - Retorna verdadeiro caso ambos operandos sejam verdadeiros OU um dos operandos seja verdadeiro; se ambos forem falsos, retorna falso.
<b>!</b>	<b>Negação</b>	!var  !true = false !false = true	(Negação lógica) Retorna falso caso o único operando possa ser convertido para verdadeiro; senão, retorna verdadeiro.

# Operadores Lógicos

```
var nome = "Paulo" // nesse exemplo os valores estão fixo
var idade = 29 // mas é legal exercitar atribuindo os valores a partir de .prompt()
var sexo = "M"
var estado = "SP"

// TESTES
var teste1 = "Eu tenho mais ou igual a 18 anos E menos de 65 para ser OBRIGADO a votar?"
var t1 = (idade >= 18 && idade < 65) // para ser true as duas precisam ser verdade

var teste2 = "Sou uma pessoa que não é diferente do sexo feminino OU moro no RJ?"
var t2 = (sexo != "F" || estado == "RJ") // para ser true, apenas uma precisa ser verdade

var teste3 = "Eu me chamo Douglas E tenho 29 anos?"
var t3 = (nome == "Douglas" && idade == 29) // para ser true as duas precisam ser verdade

document.write(teste1 + " resposta: " + t1 + "<br>" +
  teste2 + " resposta: " + t2 + "<br>" +
  teste3 + " resposta: " + t3)
```

Resultado:

Eu tenho mais ou igual a 18 anos E menos de 65 para ser OBRIGADO a votar? resposta: true  
Sou uma pessoa que não é diferente do sexo feminino OU moro no RJ? resposta: true  
Eu me chamo Douglas E tenho 29 anos? resposta: false



# Ordem de precedência dos operadores lógicos

- Ao usar os operadores lógicos, existe uma ordem de precedência, caso apareça mais de um operador na mesma linha. Então vai ser verificado a precedência superior e em seguida as demais.

!	Primordialmente a primeira ordem é a negação
&&	A segunda ordem é o E
	A terceira ordem é o OU

# Ordem de precedência entre os operadores

<b>Aritméticos</b>	Primordialmente a primeira ordem é dos operadores lógicos
<b>Relacionais</b>	A segunda ordem é dos operadores relacionais
<b>Lógicos</b>	A terceira ordem é dos operadores lógicos

# Operador Ternário

- O operador condicional é o único operador JavaScript que utiliza três operandos. O operador pode ter um de dois valores baseados em uma condição. A sintaxe é:

***condição ? valor1 : valor2***

Se condição for verdadeira, o operador terá o valor de valor1. Caso contrário, terá o valor de valor2. Você pode utilizar o operador condicional em qualquer lugar onde utilizaria um operador padrão.

Por exemplo:

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

- Esta declaração atribui o valor "adulto" à variável status caso idade seja dezoito ou mais. Caso contrário, atribui o valor "menor de idade".

# Operador Ternário

```
var nome = prompt("Qual o seu nome?")
var idade = prompt("Qual a sua idade?")
var status = (idade >= 18) ? "adulto" : "menor de idade"
document.write(nome + " tem " + idade + " anos e é " + status)
```

## Resultado:

Essa página diz

Qual o seu nome?

Essa página diz

Qual a sua idade?

← → ↺ ⓘ Arquivo | C:/Us

Paulo tem 29 anos e é adulto

Essa página diz

Qual o seu nome?

Essa página diz

Qual a sua idade?

← → ↺ ⓘ Arquivo | C:/Us

Code tem 1 anos e é menor de idade\*

*\*na verdade ele já é considerado adulto, pois é o meu **dog***

# Seleção de elementos - TAGS

- `getElementsByTagName()[]`: seleciona por TAG.

```
<body>

<h1> Estudando JavaScript </h1>
<p> É possível selecionar elementos do HTML através do <strong> JavaScript </strong> </p>

<script>
  var corpo = window.document.body //acessa o BODY do HTML
  corpo.style.background = "green" // altera o css através de .style

  var titulo = window.document.getElementsByTagName('h1')[0]
  // seleciona H1 do html e em colchetes a posição da tag que voce quer [primeiro h1 = 0] [segundo h1 = 1]
  titulo.style.color = "red"

  var paragrafo = window.document.getElementsByTagName('p')[0]
  window.document.write("<hr> No paragrafo está escrito: " + paragrafo.innerText)
  // innerText escreve apenas o texto selecionado
  window.document.write("<hr> No paragrafo está escrito: " + paragrafo.innerHTML)
  // innerHTML escreve o texto selecionado com sua formatação HTML
</script>
</body>
```

## Estudando JavaScript

É possível selecionar elementos do HTML através do JavaScript

No paragrafo está escrito: É possível selecionar elementos do HTML através do JavaScript

No paragrafo está escrito: É possível selecionar elementos do HTML através do JavaScript

# Seleção de elementos - ID

- getElementById(): seleciona por ID.

```
<body>

<h1 id="tit"> Estudando JavaScript </h1> <!-- com ID -->
<p id="par"> É possível selecionar elementos do HTML através do <strong> JavaScript </strong> </p>

<script>
    var corpo = window.document.body //acessa o BODY do HTML
    corpo.style.background = "green" // altera o css através de .style

    var titulo = window.document.getElementById('tit')
    // seleciona o ID indicado no html
    titulo.style.color = "red"

    var paragrafo = window.document.getElementById('par')
    window.document.write("<hr> No paragrafo está escrito: " + paragrafo.innerText)
    // innerText escreve apenas o texto selecionado
    window.document.write("<hr> No paragrafo está escrito: " + paragrafo.innerHTML)
    // innerHTML escreve o texto selecionado com sua formatação HTML
</script>
</body>
```

## Estudando JavaScript

É possível selecionar elementos do HTML através do JavaScript

No paragrafo está escrito: É possível selecionar elementos do HTML através do JavaScript

No paragrafo está escrito: É possível selecionar elementos do HTML através do JavaScript

# Seleção de elementos - NOME

- `getElementsByName()[ ]`: seleciona por nome.

```
<body>

  <h1 name="tit"> Estudando JavaScript </h1 <!-- com NAME -->
  <p name="par"> É possível selecionar elementos do HTML através do <strong> JavaScript </strong> </p>

  <script>
    var corpo = window.document.body //acessa o BODY do HTML
    corpo.style.background = "green" // altera o css através de .style

    var titulo = window.document.getElementsByName('tit')[0]
    // seleciona o NAME indicado no html e a posição
    titulo.style.color = "red"

    var paragrafo = window.document.getElementsByName('par')[0]
    window.document.write("<hr> No paragrafo está escrito: " + paragrafo.innerText)
    // innerText escreve apenas o texto selecionado
    window.document.write("<hr> No paragrafo está escrito: " + paragrafo.innerHTML)
    // innerHTML escreve o texto selecionado com sua formatação HTML
  </script>
</body>
```

# Seleção de elementos - CLASSE

- `getElementsByClassName()[]`: seleciona por classe.

```
<body>

<h1 class="tit"> Estudando JavaScript </h1> <!-- com CLASS -->
<p class="par"> É possível selecionar elementos do HTML através do <strong> JavaScript </strong> </p>

<script>
    var corpo = window.document.body //acessa o BODY do HTML
    corpo.style.background = "green" // altera o css através de .style

    var titulo = window.document.getElementsByClassName('tit')[0]
    // seleciona o CLASS indicado no html e a posição
    titulo.style.color = "red"

    var paragrafo = window.document.getElementsByClassName('par')[0]
    window.document.write("<hr> No paragrafo está escrito: " + paragrafo.innerText)
    // innerText escreve apenas o texto selecionado
    window.document.write("<hr> No paragrafo está escrito: " + paragrafo.innerHTML)
    // innerHTML escreve o texto selecionado com sua formatação HTML
</script>
</body>
```

## Estudando JavaScript

É possível selecionar elementos do HTML através do JavaScript

No paragrafo está escrito: É possível selecionar elementos do HTML através do JavaScript

No paragrafo está escrito: É possível selecionar elementos do HTML através do JavaScript



# Seleção de elementos - SELETOR

- querySelector: seleciona por seletor do CSS.

```
<body>

<h1 class="tit"> Estudando JavaScript </h1>
<p id="par"> É possível selecionar elementos do HTML através do <strong> JavaScript </strong> </p>

<script>
  var corpo = window.document.body //acessa o BODY do HTML
  corpo.style.background = "green" // altera o css através de .style

  var titulo = window.document.querySelector('h1.tit')
  // seletor CLASS com .
  titulo.style.color = "red"

  var paragrafo = window.document.querySelector('p#par')
  // seletor ID com #
  window.document.write("<hr> No paragrafo está escrito: " + paragrafo.innerText)
  // innerText escreve apenas o texto selecionado
  window.document.write("<hr> No paragrafo está escrito: " + paragrafo.innerHTML)
  // innerHTML escreve o texto selecionado com sua formatação HTML
</script>
</body>
```

## Estudando JavaScript

É possível selecionar elementos do HTML através do JavaScript

No paragrafo está escrito: É possível selecionar elementos do HTML através do JavaScript

No paragrafo está escrito: É possível selecionar elementos do HTML através do JavaScript

# Função

A **função** é criada a partir da palavra chave **function** seguida por:

- Nome da Função.
- Lista de argumentos para a função, entre parênteses e separados por vírgulas.
- Declarações JavaScript que definem a função, entre chaves {}.

Servem para criar uma funcionalidade que é disparada apenas após alguma ação como por exemplo o clique do mouse...

```
Function nomeDaAção(parametrosOpcionais){  
    toda funcionalidade é programada aqui dentro  
}
```

# Eventos - onclick

- onclick: na div é colocado onclick e o nome da função que será disparada.

```
<style>
  div#quadrado {
    background: yellow;
    width: 400px;
    height: 400px;
    margin: auto;
    text-align: center;
  }
</style>
</head>
<body>

  <div id="quadrado" onclick="clicar()" <!-- evento de clique -->
    <h1 class="tit"> Estudando JavaScript </h1>
    <p id="par"> Eventos do mouse no <strong> JavaScript </strong> </p>
    <p id="texto"> Algo vai acontecer... </p>
  </div>

  <script>
    var q = window.document.getElementById('quadrado')
    var t = window.document.getElementById('texto')
    function clicar(){
      // a função será disparada quando clicar na div em que tem o evento onClick com o nome da função
      t.innerText = "Clicou"
      q.style.background = "green"
    }
  </script>
</body>
```



# Eventos – onmouseenter() e onmouseout()

- Com onmouseenter() e onmouseout() é possível controlar as entradas e saídas do mouse sobre o elemento

```
<div id="quadrado" onclick="clicar()" onmouseenter="entrar()" onmouseout="sair()">
  <h1 class="tit"> Estudando JavaScript </h1>
  <p id="par"> Eventos do mouse no <strong> JavaScript </strong> </p>
  <p id="texto"> Algo vai acontecer... </p>
</div>

<script>
  var q = window.document.getElementById('quadrado')
  var t = window.document.getElementById('texto')
  function clicar(){ //onClick
    t.innerText = "Clicou"
    q.style.background = "green"
  }
  function entrar(){ // onmouseenter
    t.innerText = "Entrou"
    q.style.background = "orange"
  }
  function sair(){ // onmouseout
    t.innerText = "Saiu"
    q.style.background = "red"
  }
</script>
```

## Estudando JavaScript

Eventos do mouse no JavaScript

Entrou

## Estudando JavaScript

Eventos do mouse no JavaScript

Saiu

# Escutas - addEventListener

- Outra forma de trabalhar com ações é mandar o javascript “escutar” se teve alguma ação em determinado elemento utilizando **addEventListener()**

```
<div id="quadrado"> <!-- div agora sem os eventos direto aqui -->
  <h1 class="tit"> Estudando JavaScript </h1>
  <p id="par"> Eventos do mouse no <strong> JavaScript </strong> </p>
  <p id="texto"> Algo vai acontecer... </p>
</div>

<script>
  var q = window.document.getElementById('quadrado')
  q.addEventListener('click', clicar) // equivale ao onclick
  q.addEventListener('mouseenter', entrar) // equivale ao onmouseenter
  q.addEventListener('mouseout', sair) // equivale ao onmouseout

  var t = window.document.getElementById('texto')
  function clicar(){ // click
    t.innerText = "Clicou"
    q.style.background = "green"
  }
  function entrar(){ // mouseenter
    t.innerText = "Entrou"
    q.style.background = "orange"
  }
  function sair(){ // mouseout
    t.innerText = "Saiu"
    q.style.background = "red"
  }
</script>
```

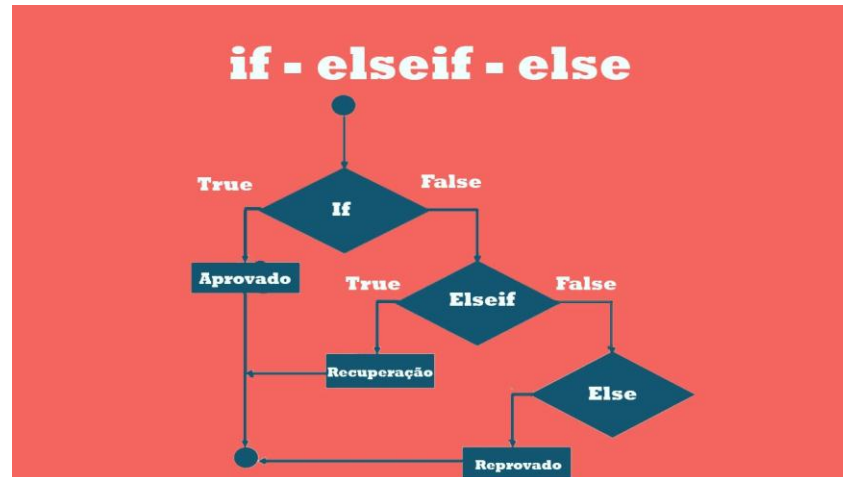
`varEscutada.addEventListener('tipoDeEvento', funcaoParaDisparar)`

- click = onclick()
- mouseenter = onmouseenter()
- mouseout = onmouseout()

Ao utilizar escuta no javascript, não se define mais o evento diretamente no HTML.

# Estrutura de Controle

- As estruturas de controle definem o fluxo de execução do programa. Elas se dividem em dois grupos: Estruturas Condicionais e Estruturas de Repetição.
- Elas controlam as instruções a serem executadas no programa.



# Estruturas de Controle

Condicionam a execução de um trecho de código, que será executado se a condição definida for verdadeira.

Temos dois tipos de estruturas condicionais:

- **If e Else**
- **Switch**



# Estrutura Condicional (if, else if, else)

A instrução **if** e **else** é utilizada para comparar duas coisas. Se caso a comparação for verdadeira **if** é executado e caso for falsa, **else** é executado.

É possível criar mais condições a partir do uso de **else if**.

## Sintaxe básica:

```
if(a==b){  
    document.write("verdadeiro")  
}  
else {  
    document.write("falso")  
}
```



# Estrutura Condicional (if, elseif, else)

A instrução **if** e **else if** tem condições, a **else** não tem e é executado caso todas condições sejam false.

```
<script>
var nota = prompt("Qual a sua nota?")
if (nota >= 7) {
    // se for maior ou igual a nota 7:
    document.write("Parabéns! Você foi aprovado!")
}
else if(nota >= 5 && nota < 7) {
    // se for maior ou igual a nota 5 E também menor 7:
    document.write("Você está de recuperação")
}
else {
    // else não tem condição (), se todas condições forem false o else executa
    document.write("Você foi reprovado")
}
</script>
```

# Estrutura Condicional (switch)

A instrução switch é similar a uma série de instruções **if**. A partir do momento que o switch encontra um valor igual ao valor da variável testada, passa a executar todos os comandos seguintes até o fim do bloco, por isso usa-se o comando **break** (quebra o fluxo finalizando a execução).

E *default* serve para executar um trecho do código que não tenha seu valor definido em case, similar ao else da estrutura if-else.

## Sintaxe básica:

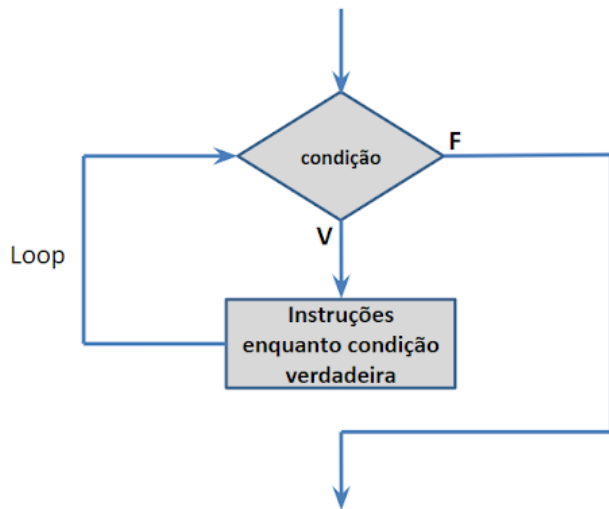
```
switch(a){  
    case "f": document.write("feminino")  
    break  
    case "m": document.write("masculino")  
    break;  
    default: document.write("outro")  
    break  
}
```

# Estrutura Condicional (switch)

```
<script>
var mes = Number(prompt("Qual número do mês atual?"))
switch(mes) {
  case 1: document.write("Estamos no mês de Janeiro!") // caso for 1:
    break
  case 2 : document.write("Estamos no mês de Fevereiro!") // caso for 2:
    break
  case 3: document.write("Estamos no mês de Março!") // caso for 3:
    break
  case 4 : document.write("Estamos no mês de Abril!") // caso for 4:
    break
  case 5: document.write("Estamos no mês de Maio!") // caso for 5:
    break
  case 6 : document.write("Estamos no mês de Junho!") // caso for 6:
    break
  case 7: document.write("Estamos no mês de Julho!") // caso for 7:
    break
  case 8 : document.write("Estamos no mês de Agosto!") // caso for 8:
    break
  case 9: document.write("Estamos no mês de Setembro!") // caso for 9:
    break
  case 10 : document.write("Estamos no mês de Outubro!") // caso for 10 ;
    break
  case 11: document.write("Estamos no mês de Novembro!") // caso for 11:
    break
  case 12 : document.write("Estamos no mês de Dezembro!") // caso for 12;
    break
  default :
    // default não tem case, caso todos cases forem false o default é executado
    document.write("Número inválido")
    break
}
</script>
```

# Estruturas de repetição

- Estruturas de repetição são recursos que permitem executar mais de uma vez trechos de código de acordo com uma condição. O JavaScript possui três estruturas de repetição: **for**, **while** e **do while**.



# Estrutura de repetição - WHILE

O **while** é o comando de repetição (loop) simples. Ele testa uma condição e executa um comando, ou um bloco de comando, até a condição testada seja falsa.

## Sintaxe básica:

```
var a=1;
while(a <=10){
    document.write(a++)
}
```

Vai imprimir na tela o número 1 e incrementar o valor, vai retornar ao loop de verificação e imprimir 2, assim até o chegar no valor igual ou maior a 10.

# Estrutura de repetição – DO WHILE

O **do while** funciona de maneira bastante semelhante ao **while**, com a simples diferença que a expressão é testada ao final o bloco de comandos.

## Sintaxe básica:

```
var a=1  
do {  
    document.write(a)  
    a++  
}  
while (a<=10)
```



# Estrutura de repetição – FOR

O **for** funciona também de maneira semelhante ao **while**, somente é organizado de uma forma diferente e simplificada.

A expressão dentro do parêntese define respectivamente:

*Inicialização da variável; Condição com a expressão a ser avaliada em cada recorrido (se for true, o loop continua); O que realizar ao final de cada loop (incremento ou decremento).*

## Sintaxe básica:

```
for(var a = 1; a <= 10; a++){  
  document.write(a)  
}
```

■ ■ ■ ■  
■ ■ ■ ■ Repare que a própria criação da variável e o incremento também ficam dentro do parêntese junto com a condição (expressão).  
■ ■ ■ ■

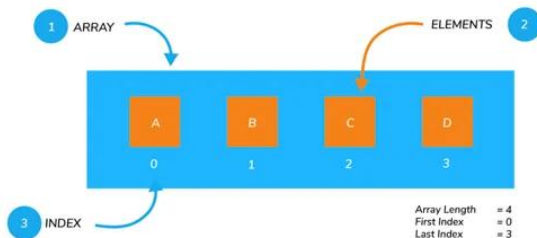
# Arrays e vetores

- Um array é um conjunto de valores ordenados que você o referencia com um nome e um índice. Por exemplo, você pode ter um array chamado emp que contém nomes de funcionários indexados por seus números de funcionários. Então emp[0] poderia ser o funcionário número 1, emp[1] o funcionário número 2 e assim por diante. As declarações a seguir criam arrays equivalentes:

```
var emp = new Array(funcionario0, funcionario1, ..., funcionarioN);
```

```
var emp= Array(funcionario0, funcionario1, ..., funcionarioN);
```

```
var emp =[funcionario0, funcionario1, ..., funcionarioN];
```





# Arrays e vetores

```
<script>
  var emp = ["funcionario1","funcionario2","funcionario3"]
  document.write(emp)
</script>
```

funcionario1,funcionario2,funcionario3

[0], [1], [2]

- Para mostrar um só, aponte a posição nos colchetes:

```
<script>
  var emp = ["funcionario1","funcionario2","funcionario3"]
  document.write(emp[1])
</script>
```

funcionario2

- Caso queira adicionar um novo elemento em um momento posterior: emp[3]="funcionario4"

```
<script>
  var emp = ["funcionario1","funcionario2","funcionario3"]
  emp[3]="funcionario4" // [3] é a posição em que vai ser inserido
  document.write(emp)
</script>
```

funcionario1,funcionario2,funcionario3,funcionario4

# Arrays e vetores – Inserção e contagem

- É possível adicionar um valor para a última posição de forma automática, sem precisar indicar a posição: `emp.push("funcionario5")`

```
<script>
  var emp = ["funcinario1","funcionario2","funcionario3"]
  emp[3]="funcionario4" // [3] é a posição em que vai ser inserido
  emp.push("funcionario5") // última posição
  document.write(emp)
</script>
```

funcinario1,funcionario2,funcionario3,funcionario4,funcionario5

- E para contar a quantidade de elementos no array: `emp.length`

```
<script>
  var emp = ["funcinario1","funcionario2","funcionario3"]
  emp[3]="funcionario4" // [3] é a posição em que vai ser inserido
  emp.push("funcionario5") // última posição
  document.write("Quantidade de posições: " + emp.length + " e os elementos são: " + emp)
```

Quantidade de posições: 5 e os elementos são: funcinario1,funcionario2,funcionario3,funcionario4,funcionario5

# Arrays e vetores - Ordenação

- É possível ordenar os valores reorganizando as posições...

```
<script>
  var emp = ["Luis","Alfredo","Ziraldo"]
  emp[3]="Bruno" // [3] é a posição em que vai ser inserido
  emp.push("Denis") // última posição
  document.write(emp)
</script>
```

Luis,Alfredo,Ziraldo,Bruno,Denis

```
<script>
  var idades = [45,78,20,5,10,8]
  document.write(idades)
</script>
```

45,78,20,5,10,8

- A partir do uso de: **sort()**

```
<script>
  var emp = ["Luis","Alfredo","Ziraldo"]
  emp[3]="Bruno" // [3] é a posição em que vai ser inserido
  emp.push("Denis") // última posição
  document.write(emp.sort())
</script>
```

Alfredo,Bruno,Denis,Luis,Ziraldo

```
<script>
  var idades = [45,78,20,5,10,8]
  document.write(idades.sort())
</script>
```

10,20,45,5,78,8

# Arrays e vetores - Loop

- Para utilizar os valores para serem mostrados na tela da forma que quiser, podendo formatá-los e adaptar o uso, basta usar um loop (for por exemplo) para trazer todos elementos. E com `.length` para contar as posições é possível flexibilizar de acordo com a quantidade de elementos.

```
<script>
  var emp = ["Luis","Alfredo","Ziraldo"]
  emp[3]="Bruno" // [3] é a posição em que vai ser inserido
  emp.push("Denis") // última posição
  for(var posicao=0;posicao<emp.length;posicao++){
    document.write("Nome do funcionário: " + emp[posicao] + "<hr>")
  }
</script>
```

---

Nome do funcionário: Luis

---

Nome do funcionário: Alfredo

---

Nome do funcionário: Ziraldo

---

Nome do funcionário: Bruno

---

Nome do funcionário: Denis

---

# Arrays e vetores – Loop com IN

- Uma maneira mais simples e moderna de realizar o mesmo procedimento é utilizar o IN, onde compreende quem, você tem uma variável para ser o contado que vai contar de acordo com a variável dos elementos (var posicao **IN** emp):

```
<script>
  var emp = ["Luis","Alfredo","Ziraldo"]
  emp[3]="Bruno" // [3] é a posição em que vai ser inserido
  emp.push("Denis") // última posição
  for(var posicao in emp){
    document.write("Nome do funcionário: " + emp[posicao] + "<hr>")
  }
</script>
```

---

Nome do funcionário: Luis

---

Nome do funcionário: Alfredo

---

Nome do funcionário: Ziraldo

---

Nome do funcionário: Bruno

---

Nome do funcionário: Denis

---

# Arrays e vetores – Posição do valor (indexOf)

- Para verificar a posição de um valor, basta utilizar o .indexOf()

```
<script>
  var emp = ["Luis","Alfredo","Ziraldo"]
  emp[3]="Bruno" // [3] é a posição em que vai ser inserido
  emp.push("Denis") // última posição
  document.write(emp.indexOf("Ziraldo"))
</script>
```

2

Posição

- Caso o valor não exista, ele retorna como -1

```
<script>
  var emp = ["Luis","Alfredo","Ziraldo"]
  emp[3]="Bruno" // [3] é a posição em que vai ser inserido
  emp.push("Denis") // última posição
  document.write(emp.indexOf("Lucas"))
</script>
```

-1

# Funções com parâmetros e retorno

- Na criação de funções, temos o parâmetros/argumentos. Eles são os valores que a função irá receber. Dentro da função temos também o **return**, que é como a função vai retornar e depois você pode decidir como usar esse retorno, seja pra fazer parte de outro calculo ou para aparecer na tela por meio de um **.write**

```
<script>
function parOuImpar(numero){
    if (numero%2==0){
        return "par"
    }
    else {
        return "impar"
    }
}
var resultado = parOuImpar(prompt("Digite um numero"))
document.write(resultado)
// ou document.write(parOuImpar(prompt("Digite um numero")))
</script>
```

Essa página diz

Digite um numero

impar

OK Cancelar

Essa página diz

Digite um numero

par

OK Cancelar

# Referências

- Mozilla: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>
- Curso em Vídeo: [https://www.youtube.com/playlist?list=PLHz\\_AreHm4dlsK3Nr9GVvXCbpOyH0I1o1](https://www.youtube.com/playlist?list=PLHz_AreHm4dlsK3Nr9GVvXCbpOyH0I1o1)

