Curso de Javascrit

Prof. Edson David Pereira

Curso de Javascrit

Onde podemos inserir JS

- Há três formas de inserir **JS** em uma página;
- **head:** link com arquivo de script;
- **body:** escrevendo códigos entre as tags script;
- **body:** link com arquivo de script.

Se for inserido dentro das tags head, para referenciar um elemento **html**, será gerado erros. O mais indicado é colocar antes do fechamento do elemento **body**, já que ele será executado posteriormente.

Curso de Javascrit

Sobre a linguagem javascript

- O código é executado de cima para baixo;
- A linguagem é case sensitive;
- Tipagem fraca;
- O ponto e vírgula a cada instrução é opcional, porém pode ocasionar erros em algumas situações. Melhor prática é inserir o ponto e vírgula, pois o padrão desde a criação era assim;
- Há comentários de uma linha e de múltiplas linhas.

Curso de Javascrit

Variáveis em javascript

- As variáveis armazenam valores;
- Estes valores podem ser utilizados posteriormente;
- No **Javascript** é possível criar variáveis de 3 formas;
- Podemos mudar o tipo da variável livremente (tipagem fraca).

```
var dado = 1;

document.write("<strong>Variáveis em Javascript</strong>")

// É possivel declarar a variável assim: dado = 'Javascript'

document.write("<p>Dado: " + dado + "</p>");

// É possível utilizar os caracteres cifrão e underline
var $linguagem = "Javascript com declaração do símbolo cifrão";
document.write("<br>" + $linguagem + "<br>");

var _linguagem = "Javascript com declaração do símbolo underline";
document.write("<br>" + _linguagem + "<br>");

// Outras formas de declarar além do comando var, é o let

let TipoLinguagem = "Estática";
const versao = 3;

// É possivel também declarar variáveis sem valor definido. Ex: var cidade;
```

Curso de Javascrit

Tipos de dados em javascript

- No **javascript** temos diversos tipos de dados, os mais comuns e utilizados são:
- Number, String, Boolean, null e undefined, object;
- Podemos verificar o tipo do dado com o operador typeof.

Curso de Javascrit

Tipos de dados em javascript

- O **Number** é para armazenar números;
- Possui três valores simbólicos: +**Infinity, -Infinity e NaN *(not a number*)**,ou seja, é uma expressão matemática que não resulta número. Esses valores não são números;
- Não existe um tipo definido para inteiros, todo número é um **Number**. Algumas linguagens separam o tipo inteiro de ponto flutuantes, o **Javascript** não é assim. Vai depender do valor armazenado, ele vai assumir determinado tipo.
- A seguir, serão utilizados os testes com os mesmos.

```
var numero = 5;

document.write(numero + '<br>');
document.write(typeof numero); // Será exibido no navegador number

var float = 5.32;
document.write("<hr>" + float + '<br>');
document.write(typeof float); // Será exibido no navegador number

var textoComNumero = "532";
document.write("<hr>" + textoComNumero + '<br>');
document.write(typeof textoComNumero); // Será exibido no navegador string

document.write(NaN);
document.write("<hr> Tipo Nan<br>" + typeof NaN); // Será exibido no navegador number

document.write("<hr>Tipo +Infinity <br>");
document.write(typeof +Infinity); // Será exibido no navegador number

document.write("<hr>Tipo -Infinity <br>");
document.write(typeof -Infinity); // Será exibido no navegador number
```

Curso de Javascrit

Tipos de dados em javascript

- **String** é para armazenar textos (letras, números sem operações aritméticas, símbolos);
- As strings podem ser escritas com as aspas simples e duplas;
- Um número entre aspas é considerado string;
- É possível também concatenar strings com o +;
- A seguir, serão utilizados os testes com os mesmos.

```
var nome = "Edson";
var sobrenome ="Pereira";

document.write("<strong>Nome: </strong>" + nome + " " + sobrenome);
document.write("<br><strong>Tipo </strong>" + typeof nome);

document.write("<hr>");
document.write("Javascript escreve 'Olá'");
```

Tipos de dados em javascript

- O **Boolean** representa um valor verdadeiro ou falso;
- Comparações resultam em booleans;
- Podemos também atribuir um valor true ou false para uma variável e ela terá o tipo **Boolean**;
- A seguir, serão utilizados os testes com os mesmos.

```
var verdadeiro = true;

document.write("<strong>Resultado: </strong>" + verdadeiro);
document.write("<br><strong>Tipo </strong>" + typeof verdadeiro);

document.write("<hr>");

var aprovado = true;
document.write("<strong>Resultado: </strong>" + verdadeiro + "<br>");
if(aprovado) {
    document.write("Aprovado");
}
else {
    document.write("Reprovado");
}
```

Curso de Javascrit

Tipos de dados em javascript

- Os tipos undefined e null também são considerados tipos de dados;
- O null é um tipo de dado que representa um valor, ou seja, imagina que abaixo desse código vai ser atribuído um valor para uma determinada variável, então inicializamos como **null *(vazio)***, para evitar possíveis erros;
- O undefined é um tipo de dado para uma variável com valor não atribuído, ou seja, a variável foi criada, porém não definido nenhum valor;
- A seguir, serão utilizados os testes com os mesmos.

Como utilizar strings

- String é o tipo de dado para textos;
- As strings podem ser escritas com aspas simples ou duplas;
- Um número entre aspas é considerado como string;
- É possível também concatenar strings com o sinal de +.

```
var nome = "Edson";
var sobrenome ="Pereira";

document.write("<strong>Nome: </strong>" + nome + " " + sobrenome);
document.write("<br><strong>Tipo </strong>" + typeof nome);

document.write("<hr>");
document.write("Javascript escreve 'Olá'");
```

Curso de Javascrit

Tipos de dados boolean em javascript

- O **Boolean** representa um valor verdadeiro ou falso;
- Comparações resultam em booleans;
- Podemos também atribuir um true ou falso para uma variável, e ela terá o tipo de

```
var verdadeiro = true;

document.write("<strong>Resultado: </strong>" + verdadeiro);
document.write("<br><strong>Tipo </strong>" + typeof verdadeiro);

document.write("<hr>");

var aprovado = true;
document.write("<strong>Resultado: </strong>" + verdadeiro + "<br>");
if(aprovado) {
    document.write("Aprovado");
}
else {
    document.write("Reprovado");
}
```

null e undefined

- **undefined e null** também são considerados tipos de dados;
- O null é um tipo de dado que representa um valor;
- O undefined é um tipo de dado para uma variável com valor não atribuído.

```
var nome = null; // inicializa
document.write("Nome: " + nome);
var sobrenome = "Pereira";
document.write("<br>Sobrenome: " + sobrenome);

nome = "Edson Pereira";
document.write("<br>Nome: " + nome);
```

Curso de Javascrit

Operadores de comparação

- São comumente utilizados para verificação nas estruturas de condição;
- Temos os seguintes operadores: ==, !=, >, <, >=, <=;
- A partir de uma expressão de comparação podemos obter um retorno **true ou false**.

```
var nome = null; // inicializa
document.write("Nome: " + nome);
var sobrenome = "Pereira";
document.write("<br>Sobrenome: " + sobrenome);

nome = "Edson Pereira";
document.write("<br>Nome: " + nome);
```

Exemplo 1:

```
<body>
    <h3>Operadores de comparação em Javascript</h3>
    <script src="js/script1.js"></script>
</body>
```

Curso de Javascrit

Operadores de comparação

Exemplo 1: exemplo1.html

```
<body>
    <h3>Operadores de comparação em Javascript</h3>
    <script src="js/script1.js"></script>
</body>
```

exemplo1.js

```
var idade = 15;
var PossuiCarro = 1;

if(idade >=18) { document.write("O usuário pode fazer a carteira"); }

if(idade <=17) { document.write("O usuário não pode fazer a carteira"); }

if(PossuiCarro) { document.write("<br>O usuário já pode andar de carro"); }

// Exemplo 1
var nome = "Edson"; // Um sinal de igual está atribuindo o valor
if(nome == "Edson") { // Dois sinal de igual está comparando (perguntando)
    document.write("<br>O nome informado é " + nome);
}

// Exemplo 2
var nome = "Marcos"; //
if(nome = "Edson") { // Um sinal de igual está afirmando
    document.write("<br>O nome informado é " + nome);
}

// Exemplo 3
var nome = "Marcos"; // Um sinal de igual está atribuindo o valor
if(nome != "Edson") { // Um sinal de exclamação e um sinal de igual está perguntando: se não for igual
    document.write("<br>O nome informado não é " + nome);
}
```

Operadores de comparação

Exemplo 2: exemplo2.html

```
<body>
    <h3>Operadores de comparação</h3>
    <form id="Operadores">
        <label>Informe a idade:</label>
        <input type="number" name="idade" id="idade" min="1" max="150" required><br>
        <label>Informe se possui carro:</label>
        <input type="checkbox" name="possuiCarro" id="possuiCarro"><br>
        <input type="submit" value="Enviar">
    </form>

    <p id="resultado">Resultado:</p>

    <script src="js/script2.js"></script>

</body>
```

Operadores de comparação

Exemplo 2:

exemplo2.js

```
document.getElementById("Operadores").addEventListener("submit", function(event) {
    event.preventDefault(); // Evita o envio do formulário

    var idade = parseInt(document.getElementById("idade").value);
    var possuiCarro = document.getElementById("possuiCarro").checked;

    if(idade >=18) {
        document.getElementById("resultado").innerHTML = "O usuário pode fazer a carteira";
    }

    if(idade <=17) {
        document.getElementById("resultado").innerHTML = "O usuário não pode fazer a carteira";
    }

    if(possuiCarro) {
        document.getElementById("resultado").innerHTML = "O usuário já pode andar de carro";
    }
});
```

Curso de Javascrit

else if e else

- Caso a instrução **if** seja negativa, podemos adicionar else;
- Que será a outra condicional a ser executada;
- Podemos então criar uma bifurcação no código;
- Já o **else if** tem a possibilidade de fazer outra verificação e adicionar mais um bloco de código

```
var nome = "Marcos";
document.write("<h4>Nome atribuído é " + nome + "</h4>");
if(nome == "Edson") {
    document.write("O nome é " + nome + "!");
} else if (nome == "Edson Júnior") {
    document.write("O nome é " + nome + " Júnior!");
} else if (nome == "Marcos") {
    document.write("O nome é Marcos!");
}
else {
    document.write("Ele possui outro nome!");
}
```

Comparação de valor e tipo de dado (=== e !==)

- Dois outros operadores de comparação existem no **Javascript:** === e !==;

unção é comparar valor e tipo de dado.

```
<body>
    <h3>Comparação de Valor e Tipo</h3>
    <script src="js/scripts.js"></script>
</body>
```

```
var numero = 5;

document.write("<h3>Número: " + numero + "</h3>");
if(numero == 5) // Compara se o número é igual a 5
{
    document.write("Este número é igual a 5!");
}

if(numero === 5) // O terceiro igual, verifica se o tipo é igual, ou seja, a variável é número inteiro, e a
comparação também
{
    document.write("<br>Este número é igual a 5!");
}

if(numero === "5") // O terceiro igual, verifica se o tipo é igual, ou seja, a variável é número inteiro, e a
comparação não irá exibir nada, pois a comparação está sendo feita com o tipo de dado que é número inteiro, e na
comparação é string
{
    document.write("Este número é igual a 5!");
}
```

Comparação de valor e tipo de dado (=== e !==)

- Dois outros operadores de comparação existem no **Javascript:** === e !==;
- A função é comparar valor e tipo de dado.

```
var numero = 5;

document.write("<h3>Número: " + numero + "</h3>");
if(numero == 5) // Compara se o número é igual a 5
{
    document.write("Este número é igual a 5!");
}

if(numero === 5) // O terceiro igual, verifica se o tipo é igual, ou seja, a variável é número inteiro, e a
comparação também
{
    document.write("<br>Este número é igual a 5!");
}

if(numero === "5") // O terceiro igual, verifica se o tipo é igual, ou seja, a variável é número inteiro, e a
comparação não irá exibir nada, pois a comparação está sendo feita com o tipo de dado que é número inteiro, e na
comparação é string
{
    document.write("Este número é igual a 5!");
}
```

Comparação de valor e tipo de dado (=== e !==)

```
var numero = '10';

document.write("<h3>Número: " + numero + "</h3>");
document.write(typeof(numero));

if(numero === 10) { // Compara se o número é igual a 10
    document.write("<br>Este número é igual a 10, e é do tipo inteiro!");
} else {
    document.write("<br>Este número é igual a 10, e é do tipo string!");
}

var numero = 15;

document.write("<h3>Número: " + numero + "</h3>");
if(numero != 10) { // Compara se o número não é igual a 10
    document.write("Este número não é igual a 10!");
}

var numero = 6;
document.write("<h3>Número: " + numero + "</h3>");
if(numero !== 8) { // Compara se o número é igual a 10
    document.write("Este número não é 16, e é do tipo number!");
}
```

Curso de Javascrit

Operador lógico AND (&&)

- Nas linguagens de programação existem os operadores lógicos;
- Estes operadores realizam comparações para que seja retornado um **true ou false**;
- Decidindo então o fluxo da aplicação;
- Utilizamos principalmente nas instruções de condição e repetição, em conjunto dos operadores de comparação;
- O operador lógico **&&** é conhecido também como **AND**;
- Ele vai retornar true apenas se as duas expressões retornarem **true**;
- Caso contrário o resultado o operador lógico **and** retornará **false**.

```
var nome = "João";
var idade = 16;

document.write("Nome: " + nome);
document.write("<br>Idade: " + idade);

document.write("<h4>Verifica se o aluno é João e a idade é igual a
16!</h4>");
if(nome == "João" && idade == 16) {
    document.write(nome + " pode entrar na aula de esgrima");
} else {
    document.write("Este não é o João");
}

var media=6;
var frequencia=75;

document.write("<br><br>Média: " + media);
document.write("<br>Frequência: " + frequencia);

document.write("<h4>Verifica se a média é maior ou igual a 6, e a
frequência maior ou igual a 75!</h4>");
if(media >=6 && frequencia >=75) {
    document.write("Aprovado");
} else {
    document.write("Reprovado");
}
```

Curso de Javascrit

Operador lógico OR (||)

- O operador lógico **||** é conhecido também como **OR**;
- Ele retorna **true** caso uma das operações retorne verdadeiro;
- O **OR** retorna false apenas se as duas expressões são

```
var nome = "João";
var idade = 16;

document.write("Nome: " + nome);
document.write("<br>Idade: " + idade);

document.write("<h4>Verifica se o aluno é João ou se a idade for maior
que 14!</h4>");
if(nome == "João" || idade > 14) {
    document.write(nome + " pode entrar na aula de esgrima");
} else {
    document.write("Não pode entrar");
}

var nota=6;
var trabalho=2;

document.write("<br><br>Nota: " + nota);
document.write("<br>Trabalho: " + trabalho);

document.write("<h4>Verifica se a nota da prova é maior ou igual a 6,
ou a nota do trabalho é maior ou igual a 6!</h4>");
if(nota >=6 && trabalho >=6) {
    document.write("Aprovado");
} else {
    document.write("Reprovado");
}
```

Operador lógico NOT (!)

- O operador lógico **!** é conhecido também como **NOT**;
- Este operador muda o valor que a expressão retornou;
- Se recebeu **true vira false**, se recebeu **false vira true**.

```
if(!false) {
   document.write("Dados estão OK");
}
```

```
if(!false) {
    document.write("<br>Aprovado");
}
else {
    document.write("<br>Reprovado");
}

var nome = "Edson";
document.write("<h3>Edson</h3>");
if(!(nome == "Ederson")) {
    document.write("O nome comparado não é Ederson!" + " e sim " + nome);
}
```

Curso de Javascrit

Estrutura de repetição while

- Estas estruturas servem para repetir **n vezes** uma operação;
- **Por exemplo:** repetir uma determinada lógica em cada elemento de um array;
- As estruturas mais comuns são: **while e for**;
- **Obs:** tomar cuidado com o loop infinito.

Exemplo 1

```
var x = 0;

document.write("<h4>Exibindo 5 números com o looping while</h4>");
while(x < 5) {
		document.write("Repetição nº " +x + "<br>");
		x+=1;
}

document.write("<h4>Exibindo array com 4 dados pessoais utilizando o looping while</h4>");
var array = ['João', 'Bebedouro', 'SP', '(17) 90909-2332'];
var y = 0;

while(y < 4) {
		document.write(" " + array[y]);
		y++;
}
```

Curso de Javascrit

Estrutura de repetição while utilizando formulário

Exemplo 2 – index.html

```
<body>
    <h3>Estrutura de repetição while</h3>
    <form id="LoopWhile">
        <label>Informe um número inteiro máximo para exibir o looping: </label>
        <input type="number" id="numero" min="1" max="10000" required>
        <input type="submit" value="Exibir">
    </form>

<div id="exibir">Exibir</div>

    <script>
        document.getElementById("LoopWhile").addEventListener("submit",
function(event) {
            event.preventDefault(); // Evita o envio do formulário

            var numero = parseInt(document.getElementById("numero").value);
            document.getElementById("exibir").innerHTML = "<h4>Exibindo " + numero + "
números com o looping while</h4>";

            var x = 0;
            var content = "";
            while (x <= numero) {
                content += "Repetição nº " + x + "<br>";
                x += 1;
            }
            document.getElementById("exibir").innerHTML += content;
        });
    </script>
</body>
```

Operadores de atribuição

- Temos algumas maneiras de atribuir um valor a uma variável;
- **As mais utilizadas são:** ++, -=, *=, /=;
- Basicamente é uma forma resumida da operação : x = x + Y:
- Mas também é comum utilizar as operadores ++ ou --.

```
var x = 1;
var y = 2;

document.write("Valor de x: " + x);
document.write("<br>Valor de y: " + y);

// Soma
document.write("<h3>Somando x + y</h3>");
x = x + y;

document.write("Valor de x: " + x);

// Subtração
document.write("<h3>Subtraindo 1 de y</h3>");
x =-y;

document.write("Valor de x: " + x);
```

Curso de Javascrit

for (estrutura de repetição - loop)

- O **for** tem uma sintaxe mais complexa, mas é optado pela maioria dos programadores;
- Apesar de parecer mais difícil, como a instrução fica toda em uma linha proporciona maior controle.

Exemplo 1:

```
for (var i = 0; i < 10; i++) {
    document.write("Estrutura de Repetição For " + i + "<br>");
}

document.write("<br>Exibindo Array<br>");
var array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];


for(var arrayIndex = 0; arrayIndex < array.length; arrayIndex++) {
    document.write(" "+ array[arrayIndex]);
}
```

Curso de Javascrit

for (estrutura de repetição - loop)

Exemplo 2:

```
<body>
    <h3>for (estrutura de repetição - loop)</h3>
    <form id="LoopFor">
        <label>Informe um número inteiro máximo para exibir o looping: </label>
        <input type="number" id="numero" min="1" max="10000" required>
        <input type="submit" value="Exibir">
    </form>

    <div id="exibir">Exibir</div>

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

```
document.getElementById("LoopFor").addEventListener("submit",
function(event) {
    event.preventDefault(); // Evita o envio do formulário

    var numero = parseInt(document.getElementById("numero").value);
    document.getElementById("exibir").innerHTML = "<h4>Exibindo" +
numero + "números com o Looping For</h4>";

    var x = 0;
    var conteudo = "";
    while (x < numero) {
        conteudo += "Repetição nº " + x + "<br>";
        x += 1;
    }
    document.getElementById("exibir").innerHTML = conteudo;
});
```

break e continue

- Com **break** podemos encerrar uma instrução;
- Com o continue podemos pular uma instrução;
- Utilizados na maioria das vezes em loops.

```
for(var i=10; i >0; i--) {
    document.write(i + " ");
    if(i === 5) {
        break;
    }
}
document.write("<h4>Break foi executado...</h4>")

var x = 10;
while(x < 100) {
    x +=10;
    if(x === 60 || x === 90) {
        // Quando o valor do x for igual a 60, ou igual a 90 ele não
exibe esse valor, e continua executando o looping
        document.write("Continue...<br>");
        continue;
    }
    document.write("Continue foi executado... " + x + "<br>")
}
```

Curso de Javascrit

funções (function)

- Funções são blocos de códigos reutilizáveis;
- Ou seja, evitamos a repetição da lógica de um programa em diversas partes do código;
- A função precisa ser invocada para ser executada.

```
function PrimeiraFuncao() {
    document.write("Alô World das funções");
}

PrimeiraFuncao();

function DizerNome(nome) {
    document.write("<br>O nome informado foi " + nome + "!");
}
DizerNome("Edson");
DizerNome("Marcos Vinicius");
DizerNome("Edson Júnior");

var NomeDoBancoDeDados = "João";
DizerNome(NomeDoBancoDeDados);

document.write("<br>");
var SomaUm = soma(2,5);
document.write(SomaUm + " <br>");
var SomaDois = soma(5, 5);
document.write(SomaDois + " <br>");
document.write(soma(4, 5) + " <br>");

function soma(a, b) {
    var soma = a + b;
    return soma;
}
```

Curso de Javascrit

Escopo em JS (scope)

- No **Javascript** podemos ter vários escopos;
- O **global**, que é iniciado em toda a aplicação;
- E os locais, que podem existir em várias instruções como as funcões.

```
var x = 1;
var y = 3;
document.write(x + " ", + " " + y);

function funcao() {
    var z = 0;
    document.write(" " + z);
}
funcao();
var  z = 10; // Atribuindo um valor, ele será exibido, senão não será exibido nada, pois o escopo será da função, definido na variável da linha 6.
document.write("<br>" + z);
```

Curso de Javascrit

let e const

- Utilizando **let e const** podemos criar escopo até em instruções como **if**;
- Deixando o código mais confiável;
- Separando cada bloco em um escopo.

```
let x = 5;

x = 12;

document.write("X = " + x);

const y = 20;
y = 21; // Será apresentado erro, pois não pode redefinir uma constant
document.write("<br>Y = " + y);


if(true)
{
    let x = 10;
    document.write("<br>X = " + x);
}

document.write("<br>X = " + x);
```

Curso de Javascrit

Métodos numéricos (parseInt, parseFloat)

- O objeto **Number**, pai dos números, contém métodos muito úteis para trabalhar com números em **JS**;
- A maioria dos tipos de dado também tem um objeto pai, como: **String, Object e Array**.

```
// parseFloat
var dado = "10.532";
document.write("<h3>Valor string: " + dado + "</h3>");
document.write("<h3>Método parseFloat</h3>");
document.write(parseFloat(dado));

console.log(dado);

// parseInt
document.write("<h3>Método parseInt</h3>");
document.write(parseInt(dado));
console.log(dado);

// toFixed
document.write("<h3>Método toFixed</h3>");
var dadoInt = parseInt(dado);
document.write(dadoInt.toFixed(2));

// isNaN
document.write("<h3>Método isNaN(not is number) - Verifica se não é
número</h3>");
document.write(isNaN(dado));

// MAX_VALUE e MIN_VALUE : Máximo valor e mínimo valor
document.write("<br>Máximo valor: " + Number.MAX_VALUE);
document.write("<br>Mínimo valor: " + Number.MIN_VALUE);
```

Curso de Javascrit

Funções de string (toUpperCase, toLowerCase, length)

- O objeto **String** também possui métodos muito úteis;
- Que vão nos auxiliar a manusear textos nos nossos

```
// length --> obtém o tamanho da string
var nome = "Linguagem Javascript";
document.write("<h3>" + nome + "</h3>");
document.write("Método length obtém o tamanho da string");
document.write("<br>" + nome + " tem " + nome.length + " caracteres");

// indexOf --> indica qual a posição que um bloco de caracteres está

document.write("<hr>Método indexOf");
document.write("<br> o texto script está " + " na posição " + nome.indexOf("script"));

// slice --> permite extrair uma string dentro de outra string
document.write("<hr>Método slice exibe em que posição encontra um determinado bloco de caracteres");
document.write("<br> o texto script está " + " na posição " + nome.slice(10, 15));

// replace
document.write("<hr>Método replace - Substituir string por outra string");
var NovaLinguagem = nome.replace("Javascript", "Python");
document.write("<br>" + NovaLinguagem);

// ToLowerCase
document.write("<hr>Método ToLowerCase - Converte para minúsculas");
var nome = nome.toLowerCase();
document.write("<br>" + nome);

// ToUpperCase
document.write("<hr>Método ToUpperCase - Converte para maiúsculas");
var nome = nome.toUpperCase();
document.write("<br>" + nome);
```

Curso de Javascrit

Funções de string (toUpperCase, toLowerCase, length)

```
// trim - elimina espaços à esquerda e à direita
document.write("<hr>Método trim - Elimina espaços à esquerda e à
direita");
var TipoLinguagem = " Linguagem dinâmica       ";
var TipoLinguagem = TipoLinguagem.trim();
document.write("<br>" + TipoLinguagem);
document.write("<br>" + TipoLinguagem + " tem " + TipoLinguagem.length
+ " caracteres");

// split
document.write("<hr>Método split - cria array");
var linguagens = "Csharp Java Python"
document.write("<br>" + linguagens.split(" "))

// lastIndexOf
document.write("<hr><h3>Método lastIndexOf</h3>");
let str = "Linguagem de Script Javascript - Utiliza blocos Script";
let TextoProcurado = "Script";
let UltimaPosicao = str.lastIndexOf(TextoProcurado);

document.write("<strong>Texto: </strong>", str);
document.write("<br><strong>Texto encontrado: </strong>",
TextoProcurado);
document.write("<br><strong>Última posição 'Script': </strong>",
UltimaPosicao);
```

Curso de Javascrit

Métodos de array – Parte 1

- Os **arrays** também possuem métodos;
- Facilitando a nossa vida para: adicionar e remover elementos, resgatar apenas uma parte do array e etc.

```
var array = [1, 2, 3, 4, 5, 6];
document.write(array.length); // Exibe quantos elementos (tamanho)
possui o array. Neste exemplo: 6

// push
document.write("<br><strong>push</strong> - adicionando elemento");
array.push(7);
array.push("Javascript");
document.write("<br>" + array);

// pop
document.write("<br><strong>pop </strong> - excluindo elemento");
array.pop();
document.write("<br>" + array);

document.write("<br><strong>unshift</strong> - adicionando elemento no
início do array");
// unshift
array.unshift(0);
document.write("<br>" + array);

document.write("<br><strong>shift - excluindo elemento no início do
array</strong>");
// shift
// shift
array.shift();
document.write("<br>" + array);
```

Curso de Javascrit

Métodos de array – Parte 2

- Podemos também resgatar um conjunto de elementos com os métodos;
- Identificar o índice de um elemento específico e muito mais.

```
// splice
var array = [1,  2, 3, 4, 5, 6, 7, 8];
document.write("<br><strong>splice</strong> - adicionando elementos");
array.splice(2, 0, 999); // Não será removido nenhum elemento, pois o
segundo parâmetro 0 (zero), indica isso, e o terceiro elemento, fará a
inserção do elemento 999.
document.write("<br>" + array);

array.splice(4, 1); // Será removido o quarto elemento(lembrando que
inicia no zero)
document.write("<br><strong>splice</strong> - removendo elementos");
document.write("<br>" + array);

// indexof
document.write("<br><strong>indexof</strong> - localiza o índice de um
determinado elemento");
document.write("<br>" + array.indexOf(5));

// join
var array2 = ["O",  "rato", "roeu", "a", "roupa"];
document.write("<br><strong>join</strong> - faz a junção de vários
blocos de string, e podemos utilizar a ,(vírgula), para separar
um  elemento do outro");
document.write("<br>" + array2.join(","));

// reverse
document.write("<br><strong>join</strong> - inverte a ordem do
array");
document.write("<br>" + array2.reverse());
```

Curso de Javascrit

Criando Objetos com métodos

- Como você pode perceber, muitos tipos de dados tem métodos e propriedades;
- Podemos criar também os nossos objetos com propriedades e métodos, para auxiliar nos nossos programas.

Curso de Javascrit

Tipos de dados Objeto

- Funcionam como um array associativo de outras linguagens;
- Podemos criar propriedades com chave e valor;
- A ideia é guardar um conjunto de valores para utilizar posteriormente.

```
var obj = {
    nome: "Edson",
    idade: 34,
    profissao: "Programador",
    estaTrabalhando: true
};

document.write("<br>Tipo de dados: " + typeof obj);
document.write("<br>Nome: " + obj.nome + "<br> Idade:" + obj.idade + "<br> Profissão: " + obj.profissao + "<br>Está trabalhando: "
+ obj.estaTrabalhando);
if(obj.estaTrabalhando)
{
    document.write("<br>Está trabalhando");
}
else
    document.write("<br>Não está trabalhando");
```

Curso de Javascrit

Utilizando o this

Fora de escopos locais o this sempre se refere ao objeto global **Window;**
Em objetos o this vai se referir a instância e pode acessar suas propriedades.

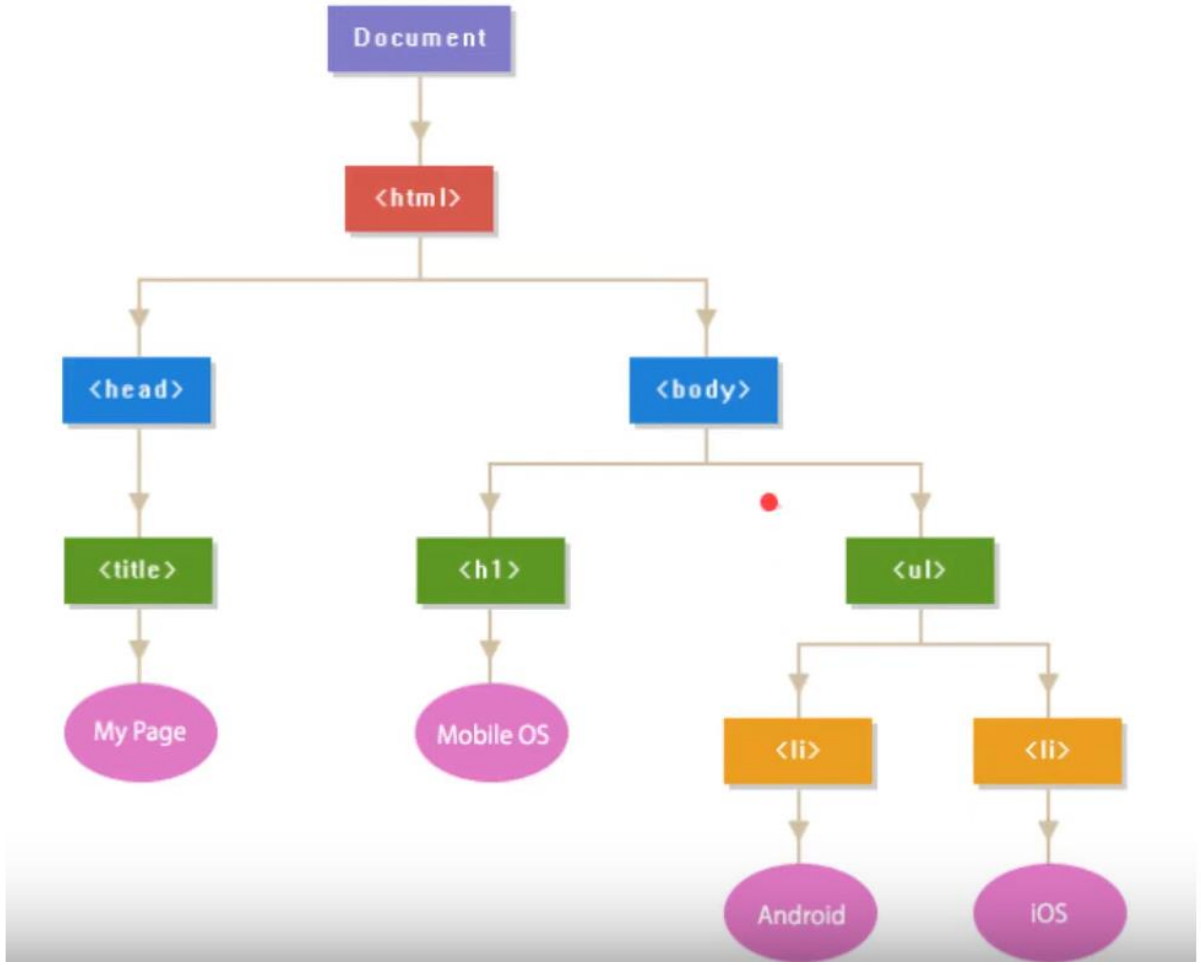
```
var numero = 10;

document.write(this); // O this representa o objeto window
console.log(this);

// Exemplos:
alert("Seja bem-vindo ao Javascript");
this.alert("Linguagem de Programação WEB!");
```

No Inspecionar, observe a lista do this:

Curso de Javascrit

O que é DOM? (Document Object Model)

- Document **Object** Model;
- Uma interface de programação para **HTML**;
- Por meio dele, temos métodos para acessar a árvore de elementos;
- O **DOM** fornece uma cópia do **HTML**;
- Podemos manipular eventos pelo **DOM** para afetar o **HTML**.



Curso de Javascrit

Acessando o DOM - Document Object Model

- Acessar o **DOM** caracteriza-se por identificar um elemento do **HTML** através de métodos;
- Depois podemos manipulá-los da forma que quisermos;
- Acessar o **DOM** é semelhante as regras de **CSS**;
- Podemos acessar por: **tags, ids, classes**.

Veja exemplo abaixo:

```
<body>
    <h3>Acessando elementos através do DOM</h3>
    <p id="paragrafo">Elemento parágrafo com id</p>
    <ul>
        <li class="item">Item 1</li>
        <li class="item">Item 2</li>
        <li class="item">Item 3</li>
        <li class="item">Item 4</li>
    </ul>
    <script src="js/script.js"></script>
</body>
```

Curso de Javascrit

Acessando o DOM - Document Object Model

```
// tag
var  titulo = document.getElementsByTagName('h3')[0];
console.log(titulo);

var lista = document.getElementsByTagName('li');
console.log(lista);

// id
var paragrafo = document.getElementById('paragrafo');
console.log(paragrafo);

// class
var itensDaLista = document.getElementsByClassName('item');
console.log(itensDaLista);
```

Acesse o navegador. No **Inspecionar / Console**, veja abaixo:

Outras formas de acessar

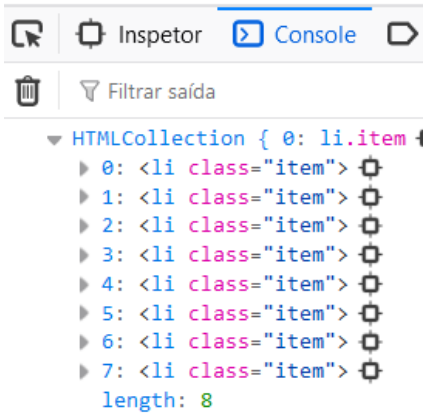
- Com a evolução da linguagem, foram criados dois seletores que acabam com toda a complexidade desta ação;
- **querySelector e querySelectorAll**;
- Com estes podemos acessar os elementos baseados em regras de **CSS**.

```
<body>
    <h3>Acessando elementos através do DOM</h3>
    <p id="paragrafo">Elemento parágrafo com id</p>
    <ul id="lista1">
        <li class="item">Item 1</li>
        <li class="item">Item 2</li>
        <li class="item">Item 3</li>
        <li class="item">Item 4</li>
    </ul>

    <ul id="lista2">
        <li class="item">Item 5</li>
        <li class="item">Item 6</li>
        <li class="item">Item 7</li>
        <li class="item">Item 8</li>
    </ul>
    <script src="js/script.js"></script>
</body>
```

Curso de Javascrit

Outras formas de acessar

```
// getElementsByClassName
var itensClasse = document.getElementsByClassName('item');
console.log(itensClasse);
```




```
// querySelectorAll
var itensQuery = document.querySelectorAll('#lista2 li');
console.log(itensQuery); // Neste exemplo, somente só serão exibidos o
id da lista2

// querySelector
var lista = document.querySelector('#lista1');
console.log(lista);
```

Curso de Javascrit

Alterar conteúdo do elemento (textContent e innerHTML)

- Podemos alterar o texto de qualquer elemento de forma fácil com **Javascript**;
- Posteriormente poderemos atrelar esta ação com algum evento.

index.html

```
<body>
    <h3 id="title">Alterar conteúdo do elemento (textContent e innerHTML)
</h3>
    <h4 class="cab4">Este é o cabeçalho 4</h4>
    <p id="paragrafo">Elemento parágrafo com id</p>
    <ul id="lista1">
        <li class="item">Item 1</li>
        <li class="item">Item 2</li>
        <li class="item">Item 3</li>
        <li class="item">Item 4</li>
    </ul>

    <ul id="lista2">
        <li class="item">Item 5</li>
        <li class="item">Item 6</li>
        <li class="item">Item 7</li>
        <li class="item">Item 8</li>
    </ul>
    <script src="js/script.js"></script>
</body>
```

script.js

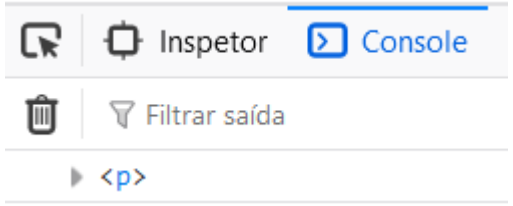
```
// selecionar elemento
var title = document.querySelector('#title');
console.log(title);

// innerHTML
title.innerHTML = 'Texto foi alterado';
// textContent -> mais utilizado, recomendado e perfomático.
var cab4 = document.querySelector(".cab4");
console.log(cab4);

cab4.textContent = 'Teste com textContent';
```

Curso de Javascrit

Criando elementos com DOM (createElement)

- Outra possibilidade do **JS** e o **DOM** é criar elementos;
- O texto de um elemento é considerado um nó na árvore do **DOM**;
- Ou seja, temos que criar o texto do elemento também.

```
var novoParagrafo = document.createElement("p");
console.log(novoParagrafo);
```



Deixe o código como abaixo:

```
var novoParagrafo = document.createElement("p");
console.log(novoParagrafo);

var texto = document.createTextNode("Este é conteúdo do parágrafo");
novoParagrafo.appendChild(texto);
console.log(novoParagrafo);

var body = document.querySelector("body");
console.log(body);

body.appendChild(novoParagrafo);
```

Criando elementos com DOM (createElement)

Este é o cabeçalho 4

Elemento parágrafo com id

- Item 1
- Item 2
- Item 3
- Item 4

- Item 5
- Item 6
- Item 7
- Item 8

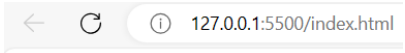
Este é conteúdo do parágrafo

Curso de Javascrit

Removendo elementos

- Remover elementos também é muito fácil com **Javascript**;
- Temos como remover o elemento diretamente e também um elemento filho.

```
<div id="container">
        <p>Testando</p>
    </div>

// removendo elemento filho
var container = document.querySelector('#container');
var p = document.querySelector('#container p');
container.removeChild(p);
```



Removendo elementos

Este é o cabeçalho 4

Elemento parágrafo com id

- Item 1
- Item 2
- Item 3
- Item 4

- Item 5
- Item 6
- Item 7
- Item 8

```
// remover o elemento
var cab4 = document.querySelector('.cab4');
cab4.remove();
```

Removendo elementos

Elemento parágrafo com id

- Item 1
- Item 2
- Item 3
- Item 4

- Item 5
- Item 6
- Item 7
- Item 8

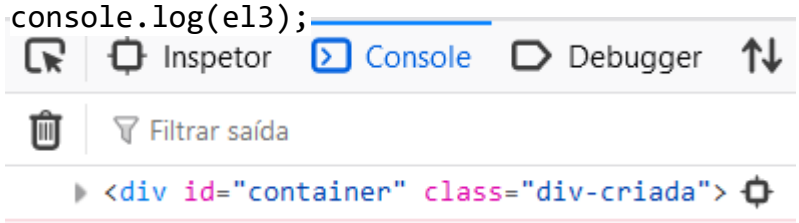
A classe cab4 foi removida.

Curso de Javascrit

Inserindo elementos (appendChild e insertBefore)

- Podemos inserir um elemento dentro de outro, por exemplo, um parágrafo em uma div;
- Isso é considerado como “acrescentar um filho” em **JS**.

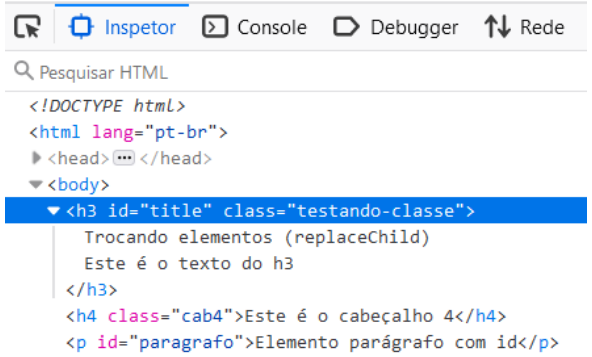
```
// criar elemento
var el = document.querySelector('div');
el.classList = "div-criada";
console.log(el);

var container = document.querySelector("#container");

// inserindo elemento filho
container.appendChild(el);

// insertBefore - insere antes
var el2 = document.createElement("div");
el2.classList = "div-before";

var el3 = document.querySelector("#container .div-criada");
console.log(el3);
```



Trocando elementos (replaceChild)

- Podemos também trocar um elemento no **DOM**;
- Ou seja, substituir uma tag por outra e de modo bem fácil.

```
// criar um elemento
var el = document.querySelector('h3');
el.classList = "testando-classe";
console.log(el);

var texto = document.createTextNode("Este é o texto do h3");
el.appendChild(texto);
console.log(el);
```

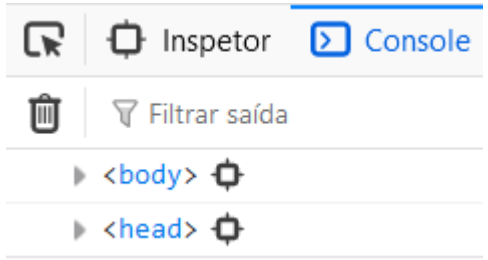



```
// criar um elemento
var el = document.querySelector('h3');
el.classList = "testando-classe";
console.log(el);

var texto = document.createTextNode(" - Este é o texto do h3");
el.appendChild(texto);
console.log(el);

// selecionar o elemento que quero trocar
var title = document.querySelector('#title');
console.log(title);

// selecionar o pai do el
var pai = title.parentNode;

// trocar os elementos
pai.replaceChild(el, title);
```

Propriedades do document

- O objeto **document** não tem somente métodos;
- Podemos retirar várias informações importantes das suas propriedades também.

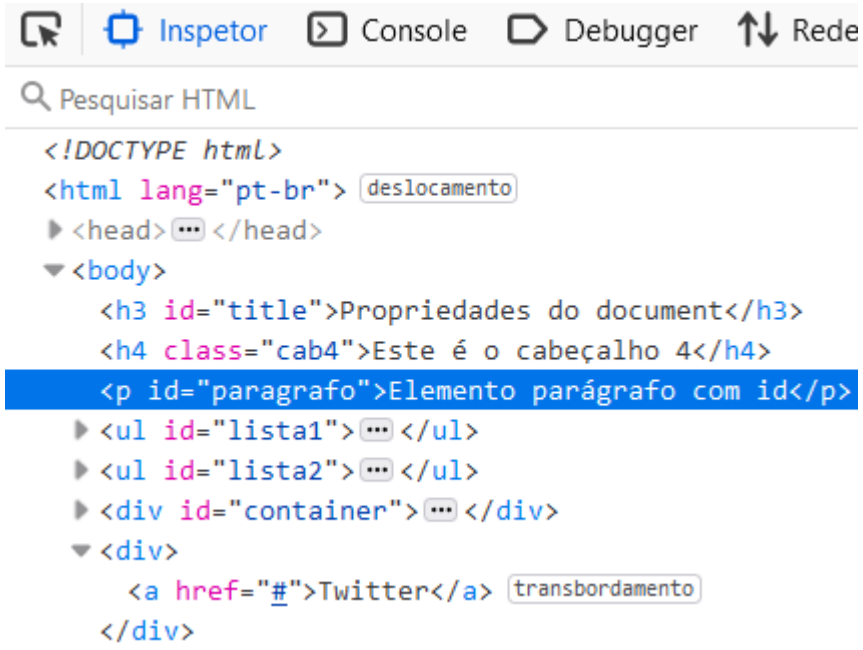
```
// propriedades document
console.log(document.body);
console.log(document.head);
```




```
// propriedades document
console.log(document.body);
console.log(document.head);
console.log(document.links);
```

Propriedades do document

```
// propriedades document
console.log(document.body);
console.log(document.head);
console.log(document.links);

console.log(document.links[0]);
document.links[0].textContent = "Twitter";
```

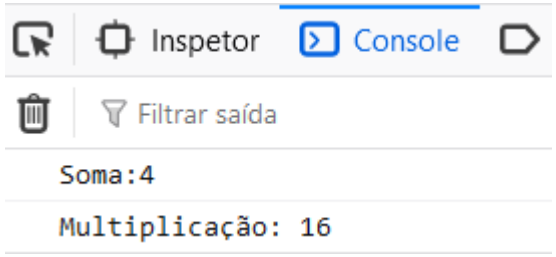


Callback functions

- A função de callback é um recurso muito interessante e também amplamente utilizado em **JS**;
- Permite executar uma função depois de uma determinada ação;
- Conceito fundamental para entender a parte assíncrona do **JS**.

Exemplo 1 (sem utilização do callback)

```
function soma(a,b) {
    var op = a + b;
    console.log("Soma:" + op);
}

function multiplicacao(a, b) {
    var op = a * b;
    console.log("Multiplicação: " + op);
}

soma(2,2);

multiplicacao(2,8);
```

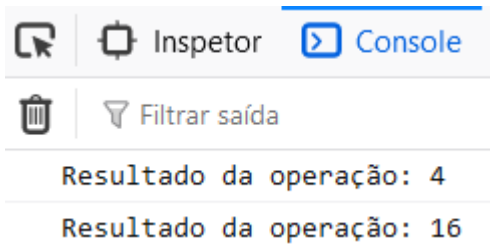


Callback functions

- A função de callback é um recurso muito interessante e também amplamente utilizado em **JS**;
- Permite executar uma função depois de uma determinada ação;
- Conceito fundamental para entender a parte assíncrona do **JS**.

Exemplo 1 (com a utilização do callback)

```
function exibir(num) {
    console.log("Resultado da operação: " + num);
}

function soma(a,b, callback) {
    var op = a + b;
    callback(op);
}

function multiplicacao(a, b, cb) {
    var op = a * b;
    cb(op);
}

soma(2,2, exibir);

multiplicacao(2, 8, exibir);
```



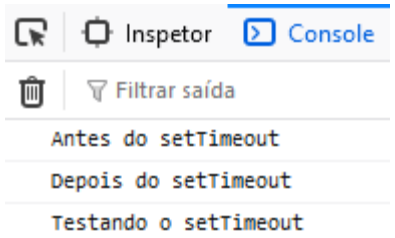
Curso de Javascrit

setTimeout e setInterval

- Podemos com estas funções criar ações no software que executam depois de um tempo ou de tempos em tempos;
- Um dos argumentos destas funções é uma callback function.

```
// setTimeout
console.log("Antes do setTimeout");
// Modo assíncrono, pois espera a resposta em 2 segundos para exibir a mensagem
setTimeout(function() {
    console.log("Testando o setTimeout");
}, 2000); // 2000 significa milisegundos (2 segundos)

console.log("Depois do setTimeout");
```

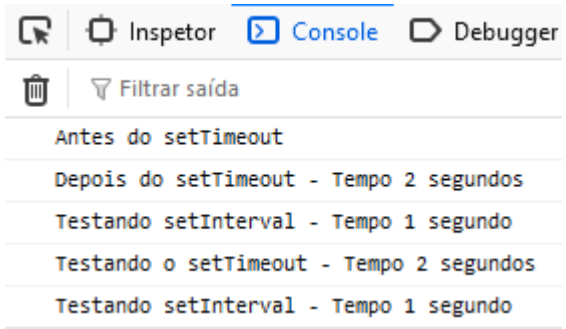
No **Inspecionar Elemento**, após 2 segundos, é exibida a mensagem abaixo:



Curso de Javascrit

setTimeout e setInterval

```
// setTimeout
console.log("Antes do setTimeout");
// Modo assíncrono, pois espera a resposta em 2 segundos para exibir a mensagem
setTimeout(function() {
    console.log("Testando o setTimeout");
}, 2000); // 2000 significa milisegundos (2 segundos)

console.log("Depois do setTimeout");

// setInterval
setInterval(function() {
                console.log("Testando setInterval");
}, 1000);
```

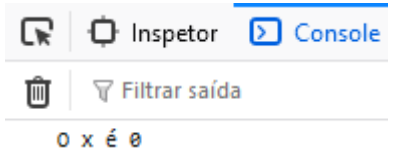


No **Inspecionar Elemento**, no exemplo acima, é executada a função do **setTimeout** que está definida 2 segundos, depois **setInterval** com 1 segundo.

clearTimeout e clearInterval

- Podemos colocar um fim em setTimeout e setInterval por meio destes dois métodos;
- Então após determinada condição os timers não serão mais executados.

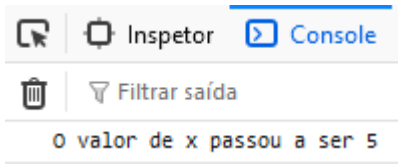
```
var x = 0;
setTimeout(function() {
		console.log("O x é 0");
}, 1500);
```



No **Inspecionar Elemento**, no exemplo acima, depois de 1 segundo e meio, é exibida a variável x = 0.

Modifique e deixe o código como abaixo:

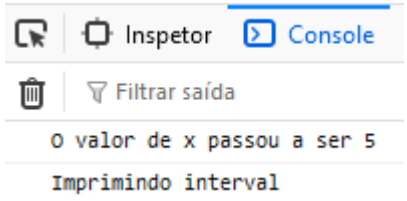
```
// clearTimeout na prática
var x = 0;
var meuTimer = setTimeout(function() {
		console.log("O x é 0");
}, 1500);

x = 5;

if(x > 0) {
		clearTimeout(meuTimer);
		console.log("O valor de x passou a ser " + x);
}
```



clearTimeout e clearInterval

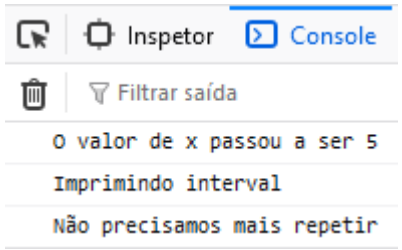
Insira a codificação abaixo:

```
// clearInterval na prática
var meuInterval = setInterval(function() {
        console.log("Imprimindo
interval");
}, 500);
```



Insira a codificação abaixo:

```
setTimeout(function(){
        console.log("Não
precisamos mais repetir");
        clearInterval(meuInterval);
}, 1500);
```



Curso de Javascrit

Eventos e onload

- Por meio de **Javascript** podemos mapear algumas ações dos usuários, que chamamos de eventos;
- **Como:** movimento do mouse, click, mouse entrando ou saindo de um elemento, carregamento da página e etc;
- **E então abrir comportamento interessante como: animação de menu abrindo e fechando.**

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Eventos e onload</title>
</head>
<body>
    <h3 id="title">Eventos e onload</h3>

    <script src="js/script.js"></script>
</body>
</html>
```

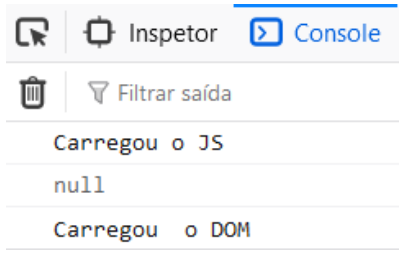
```
window.onload = function() { // Quando a página (window) carregar (load), será exibida a mensagem  abaixo
    // Observe que os scripts estão sendo carregados dentro do body, pois se for inserido dentro do head, poderá ocasionar erro,
já que os elementos (tags), que estão dentro do body, neste caso ainda não foram carregados.
    console.log("Carregou  o DOM");
}
console.log("Carregou o JS");
```

Eventos e onload

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Eventos e onload</title>
    <script src="js/script.js"></script>
</head>
<body>
    <h3 id="title">Eventos e onload</h3>
</body>
</html>
```

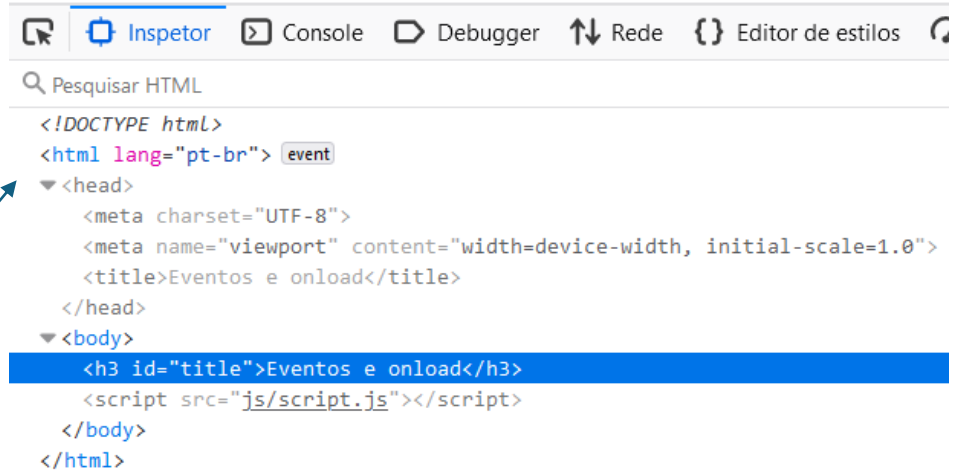
Observe que agora o script está sendo carregado dentro do head.

```
window.onload = function() { // Quando a página (window) carregar (load), será exibida a mensagem  abaixo
    // Observe que os scripts estão sendo carregados dentro do body, pois se for inserido dentro do head, poderá ocasionar erro, já que os elementos (tags), que estão dentro do body, neste caso ainda não foram carregados.
    console.log("Carregou  o DOM");
}
console.log("Carregou o JS");

var title = document.querySelector("#title");
console.log(title);
```



Observe que o elemento title está nulo (null), já que foi inserido o script antes dos elementos contidos dentro do body.

Ao lado, o script volta a ser executado antes do fechamento do elemento body. Observe que o elemento h3, é exibido com o conteúdo.



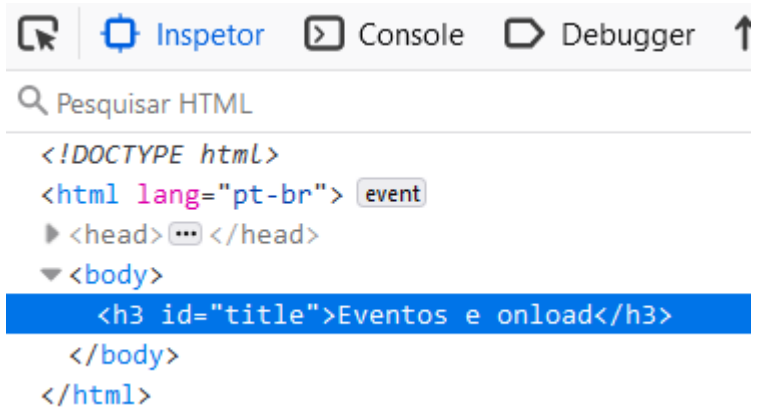
Eventos e onload

Outra forma de fazer isso, é deixar o carregamento dentro do elemento **head**.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Eventos e onload</title>
    <script src="js/script.js"></script>
</head>
<body>
    <h3 id="title">Eventos e onload</h3>
</body>
</html>
```

```
window.onload = function() { // Quando a página (window) carregar (load), será exibida a mensagem  abaixo
    // Observe que os scripts estão sendo carregados dentro do body, pois se for inserido dentro do head,
poderá ocasionar erro, já que os elementos (tags), que estão dentro do body, neste caso ainda não foram
carregados.

    console.log("Carregou  o DOM");
    var title2 = document.querySelector("#title");
    console.log(title2);
}
console.log("Carregou o JS");

var title = document.querySelector("#title");
console.log(title);
```

No script, foi definido dentro do carregamento da página, através do **querySelector**, para que o elemento h3, seja exibido. Veja ao lado:



Curso de Javascrit

Eventos click e dblclick do DOM

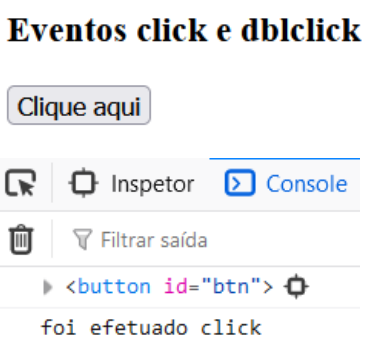
- O click é ativado quando o usuário clica em um elemento em que atrelamos o evento;
- Após a ação podemos fazer as modificações e alterações no **HTML/CSS** que quisermos.

index.html

```
<body>
    <h3 id="title">Eventos click e dblclick do DOM</h3>
    <div>
        <button id="btn">Clique aqui</button>
    </div>
    <script src="js/script.js"></script>
</body>
```

script.js

```
var btn = document.querySelector("#btn");
console.log(btn);
// addEventListener cria o evento que neste caso será o evento ckick
btn.addEventListener("click", function() {
  console.log("foi efetuado click");
});
```



Quando clica no botão **Clique aqui**, no **Inspecionar / Console**, é exibida a mensagem.

Curso de Javascrit

Eventos click e dblclick do DOM

Deixe o código como abaixo:

```
var btn = document.querySelector("#btn");
console.log(btn);
// addEventListener cria o evento que neste caso será o evento ckick
btn.addEventListener("click", function() {
  console.log("foi efetuado click");
  console.log(this); // this refere-se ao elemento button, definido no html
  this.style.color = "blue"; // Quando efetuar o click no botão, a cor será alterada
});
```

Curso de Javascrit

Eventos click e dblclick do DOM

Altere o código da página index.html abaixo:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Eventos click e dblclick do DOM</title>
</head>
<body>
    <h3 id="title">Eventos click e dblclick do DOM</h3>
   <h4 class="subtitle">Linguagem Javascript</h4>

    <div>
          <button id="btn">Clique aqui</button>
    </div>

    <script src="js/script.js"></script>
</body>
</html>
```

Altere o código do documento script.js abaixo:

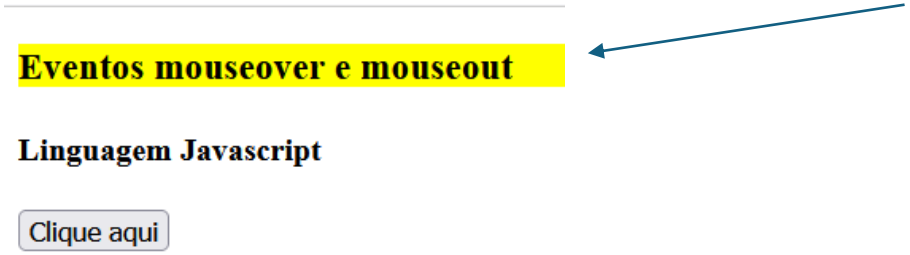
```
var title = document.querySelector("#title");
title.addEventListener("click", function() {
console.log("teste");

var subtitle = document.querySelector(".subtitle");
subtitle.style.display = "none";
})

// double click
var subtitle = document.querySelector(".subtitle");
subtitle.addEventListener("dblclick", function(){
console.log("click duplo!");
});
```

Eventos mouseover e mouseout

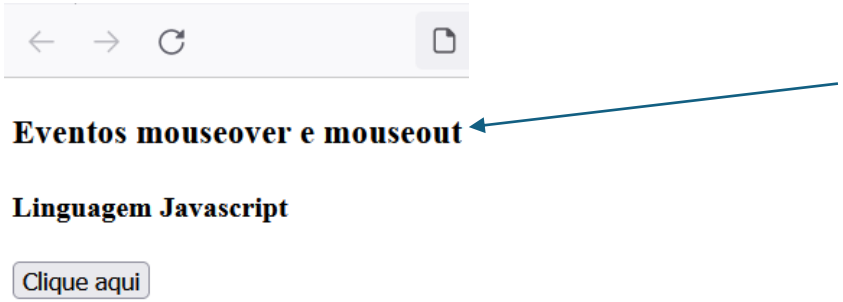
- O **mouseover** é ativado quando o ponteiro do mouse passa em cima do elemento que criamos o evento;
- Temos também o evento do **mouseout** que é quando o ponteiro sai do elemento.

```
// evento de mouseover: quando passar o mouse soubre o título (que é o elemento h3), será alterada a cor de fundo para amarelo
var title = document.querySelector("#title");

title.addEventListener("mouseover", function() {
  this.style.backgroundColor = "yellow";
});
```

Quando passar o mouse sobre o texto abaixo, e cor de fundo mouseover e mouseout, será exibida as propriedades de CSS, observe a cor de fundo.



Quando passar o mouse fora do texto abaixo, o fundo ficará com a cor branco, também através do CSS.



Curso de Javascrit

Eventos mouseover e mouseout

Altere o código da página index.html abaixo:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Eventos mouseover e mouseout</title>
    <link rel="stylesheet" href="css/estilos.css">
</head>
<body>
    <h3 id="title">Eventos mouseover e mouseout</h3>
    <h4 class="subtitle">Linguagem Javascript</h4>
    <p id="legenda" class="hide">Exemplo de evento do Javascript</p>
    <div>
        <button id="btn">Clique aqui</button>
    </div>
    <script src="js/script.js"></script>
</body>
</html>
```

Observe que entre as tags **head**, tem o link para o arquivo **estilos.css**, que está na pasta **css**. Veja abaixo:

```
h3 {
    color: green;
}

.hide {
    display: none;
}
```

Curso de Javascrit

Eventos mouseover e mouseout

Altere o código da página **script.js**, veja abaixo:

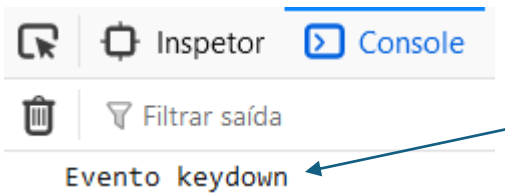
```
// evento de mouseover: quando passar o mouse soubre o título (que é o elemento h3), será alterada a cor de fundo para amarelo
var title = document.querySelector("#title");

title.addEventListener("mouseover", function() {
  this.style.backgroundColor = "yellow";
});

// evento do mouseout
title.addEventListener("mouseout", function() {
this.style.backgroundColor = "white";
});

// afetar outro elemento com mouseover
var subtitle = document.querySelector(".subtitle");

subtitle.addEventListener("mouseover", function() {
  var legenda = document.querySelector("#legenda");
  legenda.classList.remove("hide");
});
```

Eventos keydown e keyup

- O evento **keydown** é ativado quando uma tecla é pressionada;
- Podemos também atrelar o evento **keyup** para quando uma tecla volta a posição normal.

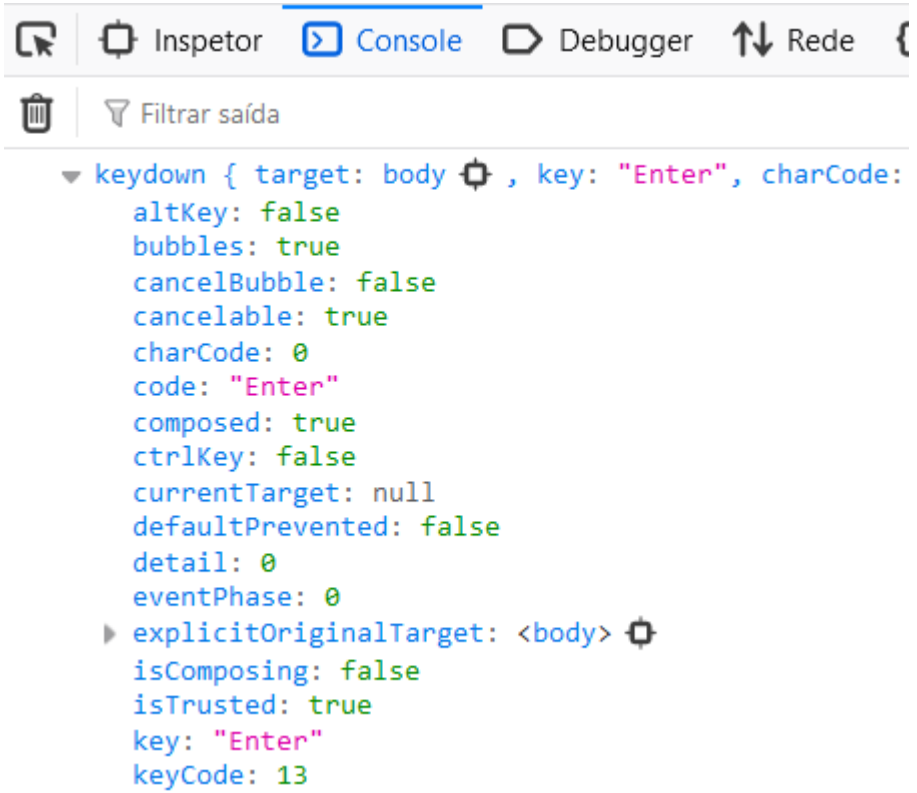
```
// keydown: este evento funciona quando pressionar qualquer
tecla
document.addEventListener("keydown", function() {
  console.log("Evento keydown");
});
```

No navegador, ao pressionar qualquer tecla, veja no **Console** abaixo a mensagem: **Eventos keydown**



Altere o código e deixe como abaixo:

```
// keydown: este evento funciona quando pressionar qualquer tecla
document.addEventListener("keydown", function(event) {
  console.log(event);
});
```



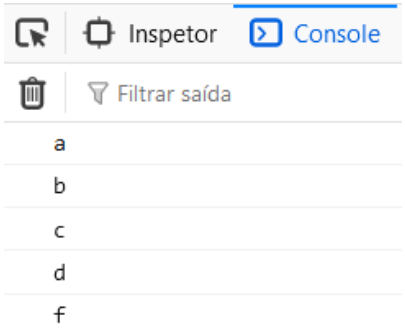
Curso de Javascrit

Eventos keydown e keyup

Altere o código e deixe como abaixo:

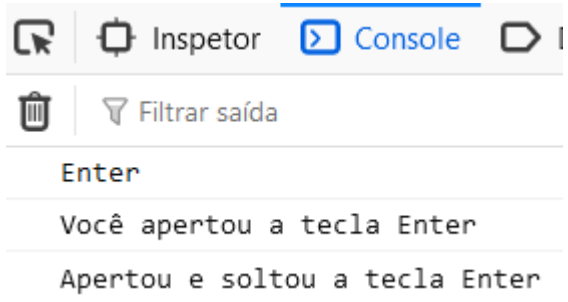
```
// keydown: este evento funciona quando pressionar qualquer tecla
document.addEventListener("keydown", function(event) {
  console.log(event.key);
});
```

No navegador, quando você pressionar por exemplo a tecla **a**, será capturada através do **event.key**, veja abaixo:



Evento **keyup**. O evento keyup abaixo, será acionado quando você soltar a tecla **Enter**. Altere o código e deixe como abaixo:

```
// keydown: este evento funciona quando pressionar qualquer tecla
document.addEventListener("keydown", function(event) {
  console.log(event.key);

  if(event.key === "Enter") {
    console.log("Você apertou a tecla Enter");
  }
});

// keyup
document.addEventListener("keyup", function(e) {
console.log("Apertou e soltou a tecla Enter");
});
```



Curso de Javascrit

Criando o projeto do curso

Acesse o site abaixo:

https://ionic.io/ionicons

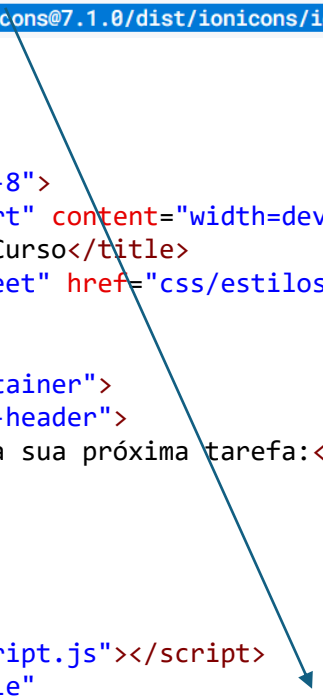
Copie o link abaixo do subtítulo **Installation**.

Installation

If you're using Ionic Framework, Ionicons is packaged by default, so no installation is necessary.

Want to use Ionicons without Ionic Framework? Place the following `<script>` near the end of your page, right before the closing `</body>` tag, to enable them.

```
' src="https://unpkg.com/ionicons@7.1.0/dist/ionicons/ionicons.esm.js"></script>
```



```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Projeto do Curso</title>
    <link rel="stylesheet" href="css/estilos.css">
</head>
<body>
    <div id="tasks-container">
        <div id="tasks-header">
            <h2>Salve a sua próxima tarefa:</h2>
        </div>
        <div>

        </div>
    </div>
    <script src="js/script.js"></script>
    <script type="module"
src="https://unpkg.com/ionicons@7.1.0/dist/ionicons/ionicons.esm.js"></script>
</body>
</html>
```

Curso de Javascrit

Criando o projeto do curso

Altere e deixe o código html como abaixo:

```
<body>
    <div id="tasks-container">
        <div id="tasks-header">
            <h2>Salve a sua próxima tarefa:</h2>
            <form action="" id="add-task-form">
                <input type="text" id="task-title" placeholder="O que você vai fazer?">
                <button id="add-btn" type="submit">
                    <ion-icon name="add-outline"></ion-icon>
                </button>
            </form>
        </div>
        <div id="tasks-list-container">
            <h2>Estas são as suas tarefas: </h2>
            <ul id="task-list">
                <li class="task-box">
                    <span class="task-title">Teste 1</span>
                    <ion-icon class="done-btn" name="checkmark-outline"></ion-icon>
                    <ion-icon class="remove-btn" name="close-outline"></ion-icon>
                </li>
                <li class="task-box">
                    <span class="task-title">Teste 2</span>
                    <ion-icon class="done-btn" name="checkmark-outline"></ion-icon>
                    <ion-icon class="remove-btn" name="close-outline"></ion-icon>
                </li>
                <li class="task-box">
                    <span class="task-title">Teste 3</span>
                    <ion-icon class="done-btn" name="checkmark-outline"></ion-icon>
                    <ion-icon class="remove-btn" name="close-outline"></ion-icon>
                </li>
            </ul>
        </div>
    </div>
    <script src="js/script.js"></script>
    <script type="module" src="https://unpkg.com/ionicons@7.1.0/dist/ionicons/ionicons.esm.js"></script>
</body>
```

Curso de Javascrit

Criando o projeto do curso

Código CSS:

```css
body {
    margin:0;
    padding:0;
    font-family: Helvetica;
    background-image: linear-gradient(to right, #e65763,#34073D);
    color: #fff;
    text-align: center;
    padding-top: 30px;
}

#tasks-container {
    width: 500px;
    margin-left: auto;
    margin-right: auto;
}

/* tasks form */
#tasks-header {
    margin-top: 40px;
}

#add-task-form {
    position: relative;
}

#add-task-form input {
    box-sizing: border-box; /* Para que os inputs não ultrapassem o limite estipulado */
    border: 1px solid #34073D;
    background: #fff;
    color: #34073D;
    padding: 15px;
    font-size: 18px;
    width: 100%;
}

#add-task-form:placeholder {
    color: #34073D;
}
```

Curso de Javascrit

Criando o projeto do curso

Código CSS:

```css
#add-task-form:focus {
    outline: none;
}

#add-btn {
    position: absolute;
    background: #fff;
    border: 1px solid #34073D;
    cursor: pointer;
    transition: .5s;
    width: 60px;
    height: 53px;
    top: 0;
    right:0;
    padding-top:5px;
}

#add-btn:hover {
    background: #34073D;
    color: #fff;
}

ion-icon {
    font-size: 25px;
}

/* tasks container */
#task-list {
    list-style: none;
    padding-left: 0;
}
```

Curso de Javascrit

Criando o projeto do curso

Código CSS:

```css
#task-list li {
    text-align: left;
    height: 50px;
    line-height: 50px;
    position: relative;
    border-bottom: 3px solid #34073D;
    background: #fff;
    color: #34073D;
    margin-bottom: 10px;
    padding-left: 15px;
    transition: .5s;
    font-weight: bold;
}

#task-list li span {
    font-size: 18px;
}

#task-list ion-icon {
    position: absolute;
    top: 0;
    height: 40px;
    transition: .5s;
    cursor: pointer;
    width: 30px;
    padding: 5px;
}

.done-btn {
    right: 40px;
}

.done-btn:hover {
    color: #fff;
    background-color: #09e018;
}
```

Curso de Javascrit

Criando o projeto do curso

Código CSS:

```css
.remove-btn {
    right: 0;
}

.remove-btn:hover {
    color: #fff;
    background-color: #eb0927;
}

.hide {
    display: none;
}

#task-list li.done {
    background-color: #34073D;
    color: #fff;
    text-decoration: line-through;
}
```

https://youtu.be/UZNUB9-tUAY?list=PLnDvRpP8BneysKU8KivhnrVaKpILD3gZ6

Curso de Javascrit

JSON

- **JSON = JavaScript Object Notation**;
- Um formato de representação de dados;
- Mais simples que **XML**, que é utilizado para fins parecidos;
- Utiliza o formato de chave e valor;
- É leve para ser enviado por requisições;
- Muito utilizado para **API** e também arquivos de configuração.

Tipos de dados

- **O JSON** aceita diversos tipos de dados;
- **Strings** – “Olá Mundo”;
- **Números** – 1 (inteiro) – 12.54 (ponto flutuante);
- **Arrays** – [1, 2, 3];
- **Objetos** - {“nome”: “Edson”;
- **Dados nulos** – null.

Curso de Javascrit

index.html

```html
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JSON</title>
    <link rel="stylesheet" href="css/estilos.css">
</head>
<body>
    <h3>JSON</h3>
    <script src="js/script.js"></script>
</body>
</html>
```

arq.json

```json
[
{
    "nome": "Edson",
    "idade": 30,
    "esta_trabalhando": true,
    "detalhes_profissao": {
        "profissao": "Programador",
        "empresa": "Empresa XYZ"
    },
    "hobbies": ["Programar", "Correr", "Ler"]
},
{
    "nome": "Elisa",
    "idade": 28,
    "esta_trabalhando": true,
    "detalhes_profissao": {
        "profissao": "Secretaria",
        "empresa": "Empresa XYZ"
    },
    "hobbies": ["Música", "Correr", "Ler"]
}
]
```

Curso de Javascrit

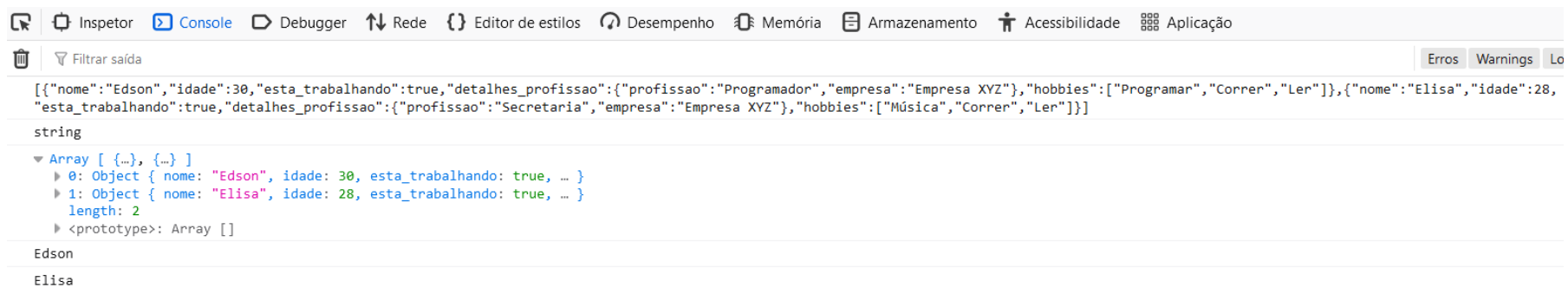
script.js

```
[
    {
        "nome": "Edson",
        "idade": 30,
        "esta_trabalhando": true,
        "detalhes_profissao": {
            "profissao": "Programador",
            "empresa": "Empresa XYZ"
        },
        "hobbies": [
            "Programar",
            "Correr",
            "Ler"
        ]
    },
    {
        "nome": "Elisa",
        "idade": 28,
        "esta_trabalhando": true,
        "detalhes_profissao": {
            "profissao": "Secretaria",
            "empresa": "Empresa XYZ"
        },
        "hobbies": [
            "Música",
            "Correr",
            "Ler"
        ]
    }
]
```

Curso de Javascrit

script.js

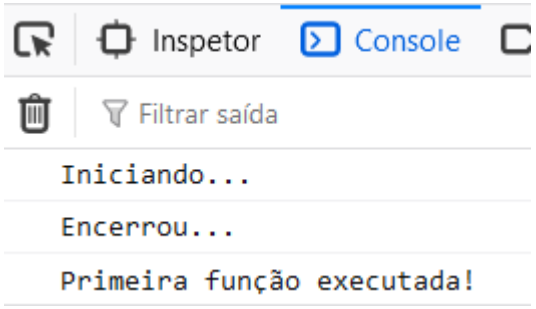
```
const objs =  [
    {
        "nome": "Edson",
        "idade": 30,
        "esta_trabalhando": true,
        "detalhes_profissao": {
            "profissao": "Programador",
            "empresa": "Empresa XYZ"
        },
        "hobbies": ["Programar", "Correr", "Ler"]
    },
    {
        "nome": "Elisa",
        "idade": 28,
        "esta_trabalhando": true,
        "detalhes_profissao": {
            "profissao": "Secretaria",
            "empresa": "Empresa XYZ"
        },
        "hobbies": ["Música", "Correr", "Ler"]
    }
]

// JSON
// converter objeto para json
const jsonData = JSON.stringify(objs);

console.log(jsonData);
console.log(typeof jsonData);

// converter json para objeto
const objData = JSON.parse(jsonData);
console.log(objData);

objData.map((pessoa) => {
    console.log(pessoa.nome);
})
```



Async e Await

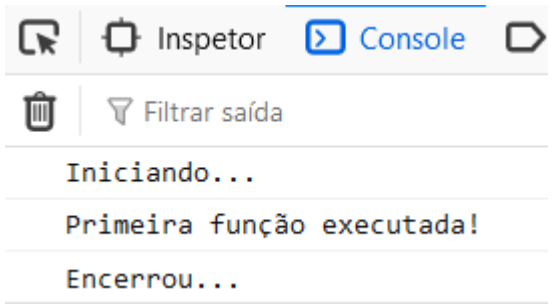
- As **funções assíncronas** funcionam como **Promises**, porém com uma sintaxe mais simples;
- Precisamos declarar a função com a palavra **async**;
- E quando precisamos aguardar por algo a instrução precisa de **await**;
- Podemos aplicar o recurso em **funções anônimas** e métodos de classe;
- Tentar usar o **await** sem o **async gera um erro**;
- Exemplo de uso: inserção de dado no banco.

```
// sintaxe
function primeiraFuncao() {
    return new Promise((resolve) => {
        setTimeout(() => {
            console.log("Primeira função executada!")
            resolve() // Entrega o resultado que está sendo esperado, no tempo de 1 segundo
        }, 1000)
    })
}

async function segundaFuncao() {
    console.log("Iniciando...")
    primeiraFuncao(); // Senão colocar o await, não será executa na ordem, ou seja: Iniciando... Primeira função executada! e depois Encerrou...

    console.log("Encerrou...");
}

segundaFuncao();
```



Async e Await

```
// sintaxe
function primeiraFuncao() {
    return new Promise((resolve) => {
        setTimeout(() => {
            console.log("Primeira função executada!")
            resolve() // Entrega o resultado que está sendo esperado, no tempo de 1 segundo
        }, 1000)
    })
}

async function segundaFuncao() {
    console.log("Iniciando...")
    await primeiraFuncao(); // Senão colocar o await, não será executa na ordem, ou seja: Iniciando... Primeira função
executada! e depois Encerrou...

    console.log("Encerrou...");
}

segundaFuncao();
```



Curso de Javascrit

getElementById

index.html

```
<h2>Digite um número de 1 a 100:</h2>

    <form id="numeroForm">
        <label for="numero">Número:</label><br>
        <input type="number" id="numero" name="numero" min="1" max="100"><br><br>
        <input type="submit" value="Enviar">
      </form>

      <p id="resultado"></p>

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

script.js

```
document.getElementById("numeroForm").addEventListener("submit", function(event) {
    event.preventDefault(); // Evita o envio do formulário

    var numeroDigitado = parseInt(document.getElementById("numero").value);

    if (numeroDigitado >= 1 && numeroDigitado <= 100) {
      if (numeroDigitado === numeroCorreto) {
        document.getElementById("resultado").innerHTML = "<p style='color:blue'>Parabéns! Você acertou o número!</p>";
      } else {
        document.getElementById("resultado").innerHTML = "<p style='color:red'>Que pena! O número correto é " + numeroCorreto + ".
Tente novamente.</p>";
      }
    } else {
      document.getElementById("resultado").innerText = "Por favor, digite um número entre 1 e 100.";
    }
  });

  var numeroCorreto = Math.floor(Math.random() * 100) + 1; // Gera um número aleatório entre 1 e 100
```

Curso de Javascrit

getElementById – Parte 2

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Verificação de Data de Agenda</title>
</head>
<body>

<h2>Agenda</h2>

<form id="agendaForm">
  <label for="dataEntrada">Data de Entrada:</label><br>
  <input type="date" id="dataEntrada" name="dataEntrada"><br><br>

  <label for="dataSaida">Data de Saída:</label><br>
  <input type="date" id="dataSaida" name="dataSaida"><br><br>

  <label for="descricao">Descrição da Agenda:</label><br>
  <textarea id="descricao" name="descricao" rows="4" cols="50"></textarea><br><br>

  <input type="submit" value="Verificar">

  <p id="resultado"></p>

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

script.js

```
document.getElementById("agendaForm").addEventListener("submit",
function(event) {
    event.preventDefault(); // Evita o envio do formulário

    document.getElementById("resultado").innerText ="";

    var dataEntrada = new
Date(document.getElementById("dataEntrada").value);
    var dataSaida = new
Date(document.getElementById("dataSaida").value);
    var descricao = document.getElementById("descricao").value;

    if (dataEntrada > dataSaida) {
      document.getElementById("resultado").innerText = "Data de
entrada é maior do que a data de saída.";
    }

    // Exibir a descrição da agenda
    document.getElementById("resultado").innerHTML += "<p
style='color:red; font-size: 20px';>Descrição da Agenda: " + descricao
+ "</p>";
    //document.getElementById("resultado").innerHTML += "<br>Descrição
da Agenda: " + descricao;
  });
```

Curso de Javascrit

index.html - Saudação

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Saudação</title>
    <link rel="stylesheet" href="css/estilos.css">
</head>
<body onLoad="alert('Seja Bem Vindo')">
    <script src="js/script.js"></script>
</body>
</html>
```

script.js

```
mdata = new Date()
mhora = mdata.getHours()
mdia = mdata.getDate()
mdiasemana = mdata.getDay()
mmes = mdata.getMonth()
mano = mdata.getFullYear()

document.write(mdiasemana);

var Mes_Extenso = new Array(13);
Mes_Extenso[1] = " de Janeiro de";
Mes_Extenso[2] = " de Fevereiro de";
Mes_Extenso[3] = " de Março de";
Mes_Extenso[4] = " de Abril de";
Mes_Extenso[5] = " de Maio de";
Mes_Extenso[6] = " de Junho de";
Mes_Extenso[7] = " de Julho de";
Mes_Extenso[8] = " de Agosto de";
Mes_Extenso[9] = " de Setembro de";
Mes_Extenso[10] = " de Outubro de";
Mes_Extenso[11] = " de Novembro de";
Mes_Extenso[12] = " de Dezembro de";
```

Curso de Javascrit

```
var time = new Date();
var lmonth = Mes_Extenso[time.getMonth() + 1];
var date = time.getDate();
var year = time.getFullYear();

document.write("<b><center>" + date + lmonth + " ");
document.write(" " + year + "</center></b>");
var Exibir_Mensagem = '';
if (mhora < 12)
    Exibir_Mensagem = "Bom dia";
else if (mhora >= 12 && mhora < 18)
    Exibir_Mensagem = "Boa tarde";
else if (mhora >= 18 && mhora < 24)
    Exibir_Mensagem = "Boa noite";
document.write("<p>" + Exibir_Mensagem + "</p>");
```

Curso de Javascrit

index.html - Saudação

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Saudação em Javascript</title>
</head>
<body>
    <script src="js/script.js"></script>
</body>
</html>
```

```
var data = new Date(); //armazena a data e a hora nesta variável
var hora = data.getHours(); //armazena a hora
var minuto = data.getMinutes(); //armazena os minutos
var segundo = data.getSeconds(); //armazena os segundos
//Exibe a hora atual
document.write("<p align='center'>Agora são " + hora + " horas, "  +
minuto + " minutos" + " e " + segundo + " segundos</p>");
var dia = data.getDate(); //armazena o dia
var ano = data.getFullYear(); //armazena o ano
var mes_extenso="";
var mes = data.getMonth(); //Armazena o mês
if (mes ==0) mes_extenso="Janeiro";
if (mes ==1) mes_extenso="Fevereiro";
if (mes ==2) mes_extenso="Março";
if (mes ==3) mes_extenso="Abril";
if (mes ==4) mes_extenso="Maio";
if (mes ==5) mes_extenso="Junho";
if (mes ==6) mes_extenso="Julho";
if (mes ==7) mes_extenso="Agosto";
if (mes ==8) mes_extenso="Setembro";
if (mes ==9) mes_extenso="Outubro";
if (mes ==10) mes_extenso="Novembro";
if (mes ==11) mes_extenso="Dezembro";
```

```
//Exibe a data atual
document.write("<p align='center'>" + dia + " de " + mes_extenso + " de " + ano + "</p  >");
var Data_Extenso = "";
if (hora <12)
    Data_Extenso = "Bom dia";
else
if (hora <18)
Data_Extenso = "Boa tarde";
else
Data_Extenso = "Boa noite";
document.write("<p align='center'>" + Data_Extenso + "</p>");
```