
JavaScript (JS)

Introdução

— Desenvolvimento Web Básico —

Base do *front-end*

HTML: Linguagem de marcação

CSS: Linguagem de estilo

JavaScript: Linguagem de programação

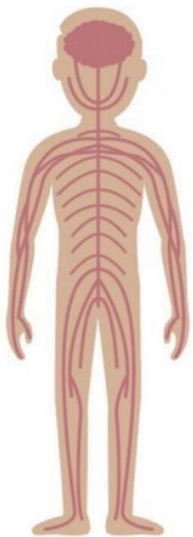
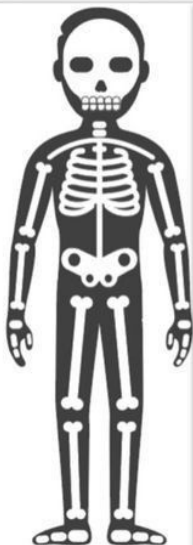


Usos

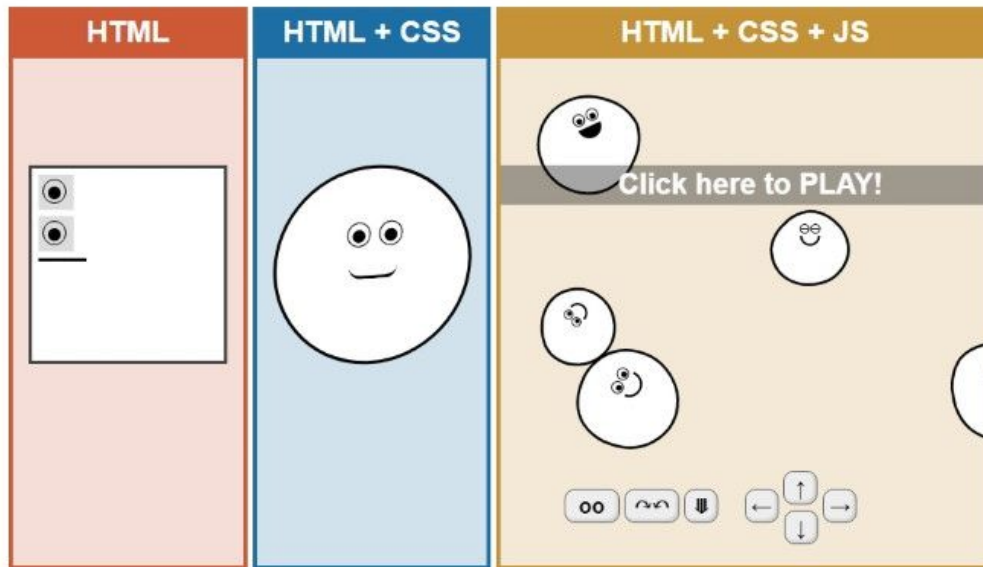
HTML

JS

CSS



<https://qph.cf2.quoracdn.net/main-qimg-6795e6ef1ca101a38e86eee75ed1189f-lq>



<https://html-css-js.com/images/og.jpg>

Inserção na página HTML

O código JavaScript pode ser inserido em uma página web de duas maneiras:

- Entre as tags `<script></script>`

```
<script>  
  
    // Código JS  
  
</script>
```

- Importado de um arquivo externo
 - Arquivos contendo código JavaScript terminam com a extensão **.js**

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

Inserção na página HTML

As tags script podem ser inseridas tanto no **head** quanto no **body** da página:

```
<script src="meu_script_1.js"></script>

<title>Document</title>

</head>
<body>

  <script src="meu_script_2.js"></script>

  <p>Olá</p>

  <script src="meu_script_3.js"></script>
</body>
```

Exercício (1)

Vamos criar nosso primeiro Hello World!

Crie uma página com o seguinte código:



```
<meta name="viewport" content="width=device-width,  
<script>  
  
    console.log("Hello World!");  
  
</script>  
  
<title>Document</title>
```

Exercício (2)

Vamos criar nosso primeiro Hello World!

Passe o código para o arquivo **meu_script.js** e vincule ao seu **index.html**

```
JS meu_script.js
1  console.log("Hello World!");
2
```

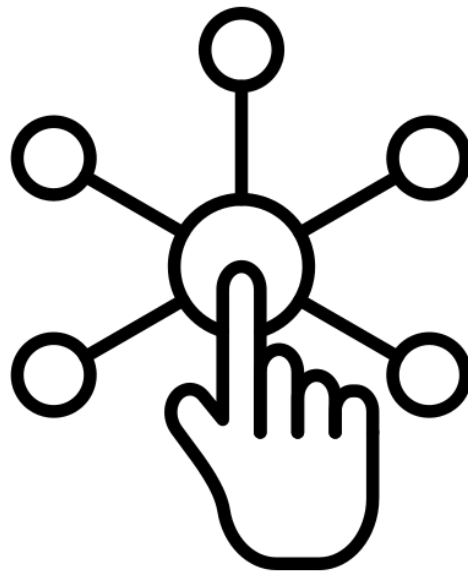
```
<> index.html >  html >  head
/
8      <script src="meu_script.js"></script>
9
```

Interações com a página

Interação com a página

O JavaScript pode apresentar informações de diferentes maneiras:

- Escrevendo no console
- Escrevendo na página
- Mostrando mensagens
- Alterando um elemento da página

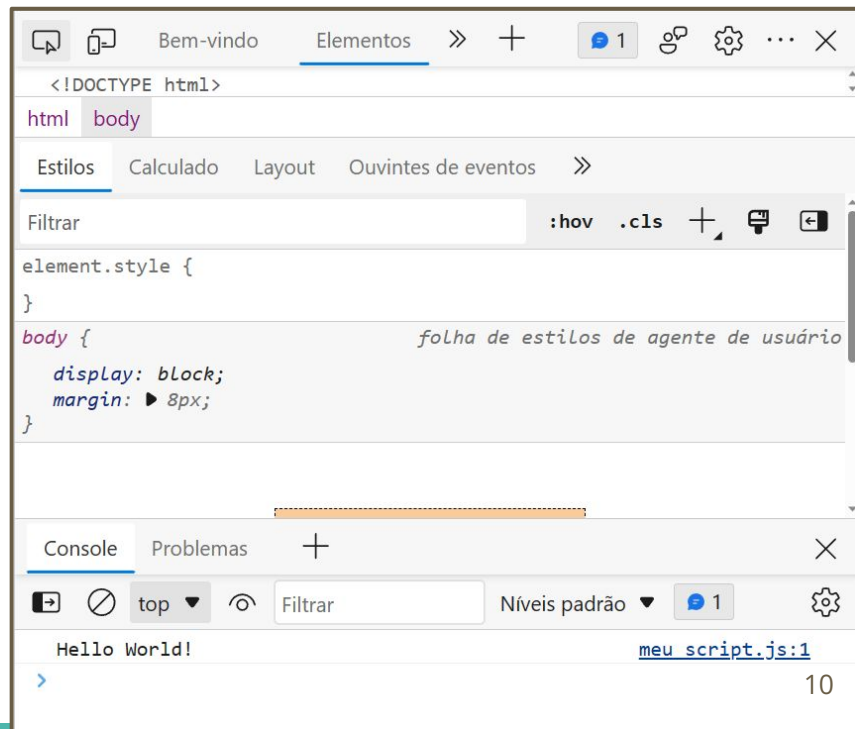


<https://cdn-icons-png.flaticon.com/512/6497/6497682.png>

Interação com a página

O JavaScript pode apresentar informações de diferentes maneiras:

- **Escrevendo no console**
- Escrevendo na página
- Mostrando mensagens
- Alterando um elemento da página

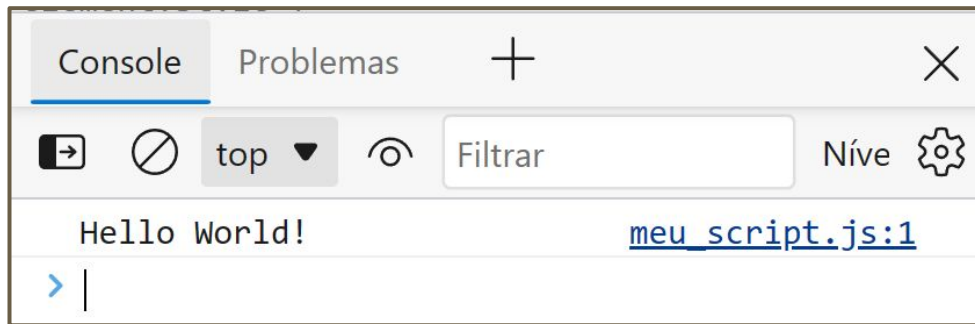


Escrever no console

A maneira mais simples de apresentar informações (principalmente para depuração) é escrevendo no console.

O console pode ser acessado por meio da tecla **F12**, ou inspecionando a página.

```
JS meu_script.js
1 console.log("Sou o JS");
2
```

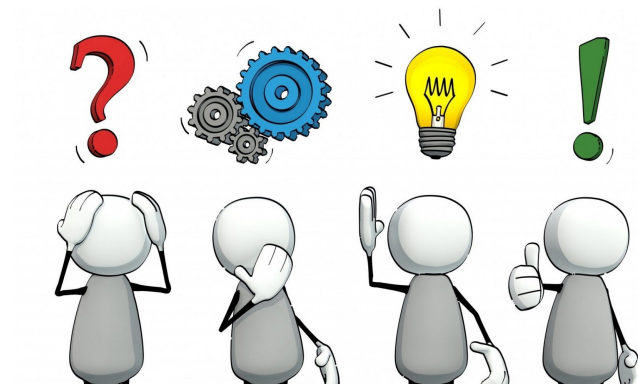


Exercício

Escreva as seguintes 3 mensagens no console:

- Bem vindo!
- Nosso site está aberto
- $17 + 5$

```
JS meu_script.js  
1 console.log("Sou o JS");  
2
```



https://www.incimages.com/uploaded_files/image/1920x1080/getty_506903004_200013332000928076_348061.jpg

Interação com a página

O JavaScript pode apresentar informações de diferentes maneiras:

- Escrevendo no console
- **Escrevendo na página**
- Mostrando mensagens
- Alterando um elemento da página



<https://smarkiss.net/wp-content/uploads/Facebook-7-tecnicas-para-escrever-os-melhores-posts.jpg>

Escrever na página

Durante o carregamento de uma página, podemos adicionar informações a ela

Porém, após a página ter sido carregada, este método apaga todo seu conteúdo

- Por conta disso, recomenda-se o uso apenas para testes...

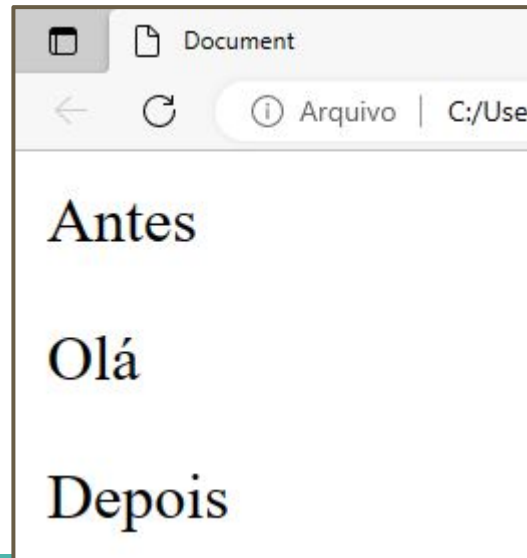
```
<body>

  <script>document.write("Antes");</script>

  <p>Olá</p>

  <script>document.write("Depois");</script>

</body>
```



```
<script>document.write("Exemplo");</script>
```

Exercício

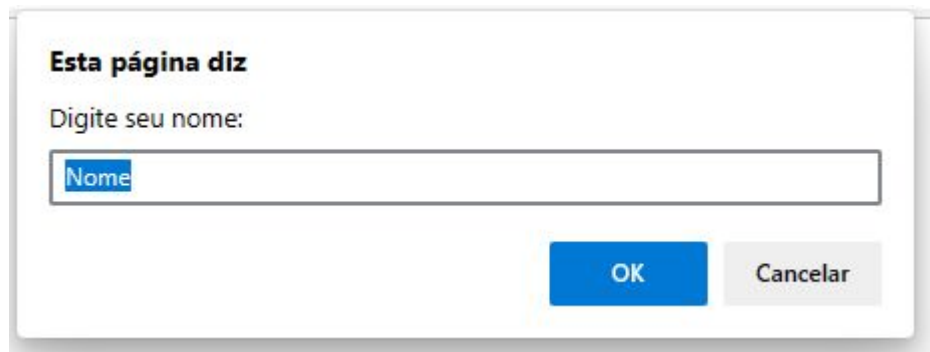
Dado o seguinte código HTML, utilize JS para completar com um valor de resultado:

```
<body>  
  
    <p>O resultado de 5 - 3 é: </p>  
  
</body>
```

Interação com a página

O JavaScript pode apresentar informações de diferentes maneiras:

- Escrevendo no console
- Escrevendo na página
- **Mostrando mensagens**
- Alterando um elemento da página



Esta página diz

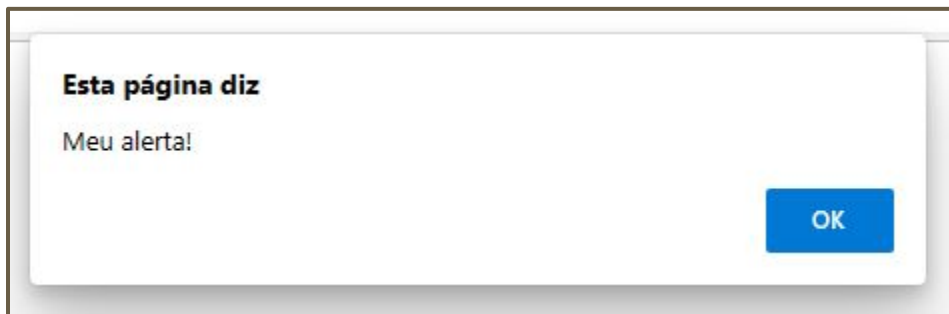
Digite seu nome:

OK Cancelar

Mostrando mensagens

Outro recurso para interagir com o usuário é por meio de mensagens, como o alerta:

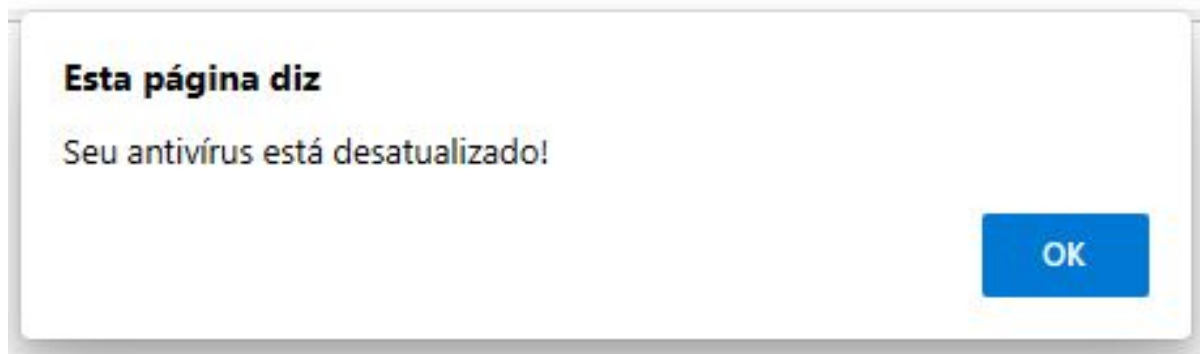
```
JS meu_script.js  
1  alert("Meu alerta!");  
2
```



Exercício

```
JS meu_script.js  
1   alert("Meu alerta!");  
2
```

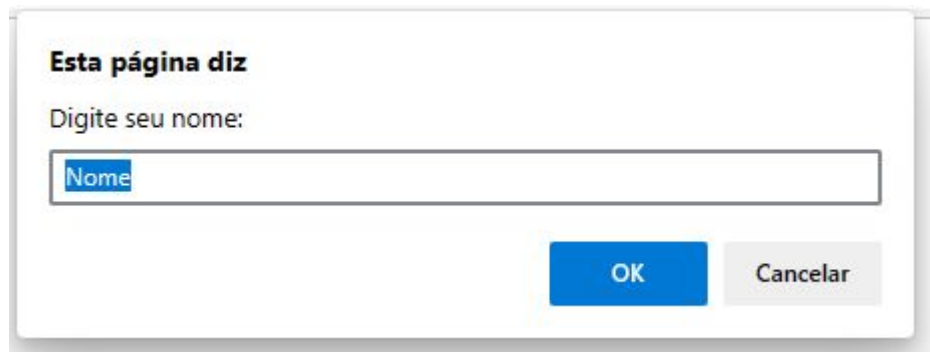
Exiba um alerta recepcionando o usuário em sua página!



Interação com a página

O JavaScript pode apresentar informações de diferentes maneiras:

- Escrevendo no console
- Escrevendo na página
- Mostrando mensagens
- **Alterando um elemento da página**



Esta página diz

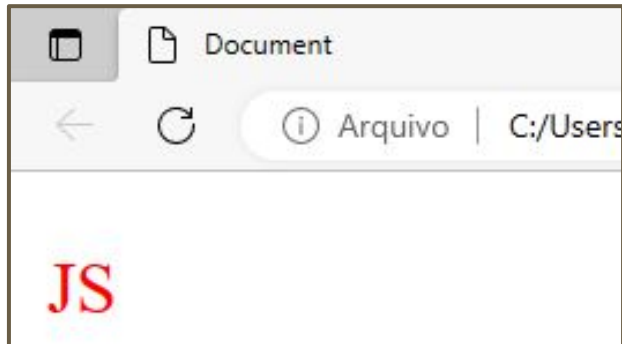
Digite seu nome:

OK Cancelar

Mostrando mensagens

Por fim, o mais interessante, podemos alterar o conteúdo e a formatação de elementos de nossa página WEB!

```
JS meu_script.js
1  document.querySelector("p").innerHTML = "JS";
2  document.querySelector("p").style.color = "red";
```



```
document.querySelector("p").style.color = "grey";
```

Exercício

Dada a seguinte página WEB, utilize JS para deixar a descrição da imagem verde com fundo cinza.

```
<body>  
  
      
  
    <p>Belo lago</p>  
  
</body>
```

Ordem de inclusão

Ordem de execução

Os elementos de uma página WEB são identificados de cima para baixo

Assim, a posição do seu JavaScript em relação aos outros elementos da página pode alterar seu resultado!

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

```
<p>Elemento</p>
```

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

Ordem de execução

Isso pode levar a problemas, como não encontrar um elemento que deveria existir na página!

Por conta disso, scripts externos podem ser carregados em diferentes modos:

- **Padrão:** é baixado e executado na ordem em que se encontra no documento.
- **Async:** é baixado em paralelo a outras partes da página. Quando o download está completo, é executado.
- **Defer:** é baixado em paralelo a outras partes da página. Porém só é executado quando a página já foi renderizada (após o “carregamento” de tudo).

Ordem de execução

- Na dúvida? **Defer** evita possíveis problemas!

```
<meta name="viewport" content="width=device-w  
  
<script src="meu_script.js" defer></script>  
  
<title>Document</title>
```

Ordem de execução

- Na dúvida? **Defer** evita possíveis problemas!

Mesmo estando no head, executa apenas quando a página termina de ser renderizada.

```
<meta name="viewport" content="width=device-w
<script src="meu_script.js" defer></script>
<title>Document</title>
```

Exercício

Use o defer para importar um arquivo JS no head que selecione a primeira div da página e altere seus estilos para:

- width: 400px;
- height: 300 px;
- backgroundColor: yellow;

```
<script src="meu_script.js" defer></script>
</head>
<body>

    <div></div>

</body>
```

Comentários

Comentários

Diferentemente do HTML e do CSS, em JS temos dois tipos de comentário:

```
// Comentário de uma única linha
```

```
/*
```

```
Comentário
```

```
em múltiplas
```

```
linhas
```

```
*/
```

Variáveis

Variáveis

Existem 3 maneiras de declarar uma variável em JavaScript (JS)

- **var** - escopo global, não delimitada por blocos (chaves) - Pode ser redeclarada
- **let** - escopo local, é válida apenas dentro do bloco onde foi declarada (chaves) - Não pode ser redeclarada
- **const** - Escopo local, com valor fixo, não pode ser alterada - Não pode ser redeclarada

Exemplo - var

```
var a = 0;  
// Valor de a: 0  
  
var a = 1;  
// Valor de a: 1  
  
a = 2;  
// Valor de a: 2  
  
{  
    var a = 3;  
    // Valor de a: 3  
}  
  
// Valor de a: 3
```


Exemplo - let

```
let a = 0;
// Valor de a: 0

// let a = 1; // Não posso redeclarar
//           // no mesmo escopo

a = 2;
// Valor de a: 2

{
    let a = 3;
    // Valor de a: 3
}

// Valor de a: 2
```

Exemplo - let

```
let a = 0;
// Valor de a: 0

// let a = 1; // Não posso redeclarar
//           // no mesmo escopo

a = 2;
// Valor de a: 2

{
    let a = 3;
    // Valor de a: 3
}

// Valor de a: 2
```

Esse a = 3 só
existe neste
escopo.

Exemplo - const

```
const a = 0;
// Valor de a: 0

// const a = 1; // Não posso redeclarar
//           |           |           // no mesmo escopo
//
// a = 2;       // Não posso alterar o valor

{
    const a = 3;
    // Valor de a: 3
}

// Valor de a: 0
```

Exemplo - const

```
const a = 0;
// Valor de a: 0

// const a = 1; // Não posso redeclarar
//           |           |           // no mesmo escopo

// a = 2;       // Não posso alterar o valor

{
    const a = 3;
    // Valor de a: 3
}

// Valor de a: 0
```

Esse a = 3 só
existe neste
escopo.

Regra geral

Na dúvida, siga a seguinte regra:

- O valor precisa mudar durante a execução?
 - **Sim** - **let**
 - **Não** - **const**



<https://nova-escola-producao.s3.amazonaws.com/atfjHXYX32AAZEWwjgkvq7HREsNDyzfEN2ACQZsruP66qszaBrVX8VvA7n4/shutterstock-576831568.jpg>

```
document.querySelector("p").innerHTML = x;
```

Exercício

Crie duas variáveis **a** e **b** e duas constantes **c** e **d**, contendo os valores 1, 2, 3 e 4, respectivamente.

Altere o parágrafo (**p**) da página para conter a soma entre esses valores:

```
<script src="meu_script.js" defer></script>
</head>
<body>

  <p></p>

</body>
```

Tipos de dados

Tipos de dados

Apesar de não definirmos explicitamente, uma variável em JS assume um dentre diversos tipos existentes na linguagem.

Alguns dos mais relevantes são:

- String
- Number
- Boolean
- Undefined
- Null
- Object

Tipos de dados

Apesar de não definirmos explicitamente, uma variável em JS assume um dentre diversos tipos existentes na linguagem.

Alguns dos mais relevantes são:

- **String**
- Number
- Boolean
- Undefined
- Null
- Object

```
let a = "Hello World!";
```

Tipos de dados

Apesar de não definirmos explicitamente, uma variável em JS assume um dentre diversos tipos existentes na linguagem.

Alguns dos mais relevantes são:

- String
- **Number**
- Boolean
- Undefined
- Null
- Object

```
let a = 42;
```

Tipos de dados

Apesar de não definirmos explicitamente, uma variável em JS assume um dentre diversos tipos existentes na linguagem.

Alguns dos mais relevantes são:

- String
- Number
- **Boolean**
- Undefined
- Null
- Object

```
let a = false;
```

Tipos de dados

Apesar de não definirmos explicitamente, uma variável em JS assume um dentre diversos tipos existentes na linguagem.

Alguns dos mais relevantes são:

- String
- Number
- Boolean
- **Undefined**
- Null
- Object

let a; **// Valor antes de inicializar**

Tipos de dados

Apesar de não definirmos explicitamente, uma variável em JS assume um dentre diversos tipos existentes na linguagem.

Alguns dos mais relevantes são:

- String
- Number
- Boolean
- Undefined
- **Null**
- Object

let a = **null**; // Representa o vazio

Tipos de dados

Apesar de não definirmos explicitamente, uma variável em JS assume um dentre diversos tipos existentes na linguagem.

Alguns dos mais relevantes são:

- String
- Number
- Boolean
- Undefined
- Null
- **Object**

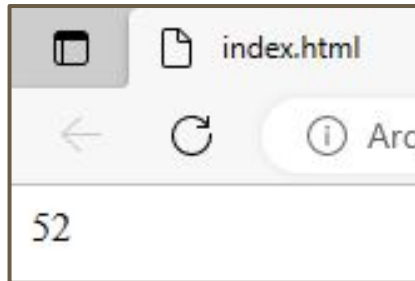
```
let Cachorro = {nome: "Mel", idade: 7, cor: "Amarela"}
```

```
<script>document.write("Exemplo");</script>
```

Exercício (1)

Crie as variáveis **a** e **b**. Coloque o número 42 em **a** e 10 em **b**.

Mostre na página WEB o resultado da soma entre **a** e **b**.



```
<script>document.write("Exemplo");</script>
```

Exercício 2

Crie o objeto **computador** com os atributos {cpu: "i7-4790", dram: "16 GB"}

- Mostre os dados do computador
 - Podemos acessar utilizando **computador.cpu** e **computador.dram**

```
JS meu_script.js
```

```
1 document.querySelector("p").innerHTML = "JS";  
2 document.querySelector("p").style.color = "red";
```


Operadores

Operadores aritméticos

Operador	Ação	Exemplo
+	Soma Concatenação	$2 + 3 == 5$
-	Subtração	$2 - 3 == -1$
*	Multiplicação	$2 * 3 == 6$
/	Divisão	$2 / 3 == 0.666\dots$
%	Módulo (resto da divisão)	$2 / 3 == 2$
**	Exponenciação (elevar a)	$2 ** 3 == (2 * 2 * 2) == 8$
++	Incremento (+1)	a++ igual (a = a + 1)
--	Decremento (-1)	a-- igual (a = a - 1)

Efeitos inesperados

Observe que o efeito da operação `+` é diferente para `Number` e `String`!

- `"Oi" + " tudo bem" == "Oi tudo bem"`
- `4 + 7 == 11`
- `"4" + "7" == "47"`
- `"4" + 7 == "47"`
- `"4" + 1 + 8 == "418"`
- `1 + 8 + "4" == "94"`

Efeitos inesperados

Observe que o efeito da operação `+` é diferente para `Number` e `String`!

- `"Oi" + " tudo bem"` == `"Oi tudo bem"`
- `4 + 7` == `11`
- `"4" + "7"` == `"47"`
- `"4" + 7` == `"47"`
- `"4" + 1 + 8` == `"418"`
- `1 + 8 + "4"` == `"94"`



Quando mistura, converte para string a partir da primeira string

```
document.querySelector("p").innerHTML = x;
```

Exercício

- Crie as variáveis **a** e **b**.
- Atribua os valores **"Resultado: "** e **42** a essas variáveis.
- Crie uma variável **c** com o resultado da soma (concatenação) dessas variáveis.
- Mostre esse resultado no parágrafo da página:

```
<script src="meu_script.js" defer></script>
</head>
<body>

    <p></p>

</body>
```

Operadores de comparação

Operador	Ação	Exemplo
==	Igual em valor	"2" == 2 (true)
===	Igual em valor e tipo	"2" === 2 (false)
!=	Diferente em valor	"2" != 3 (true)
!==	Diferente em valor ou tipo	"2" !== 2 (true)
>	Maior que	2 > 3 (false)
<	Menor que	2 < 3 (true)
>=	Maior ou igual	3 >= 3 (true)
<=	Menor ou igual	2 <= 3 (true)

Operadores lógicos

Os operadores lógicos presentes em JavaScript são similares a C e Java:

Operador	Ação	Exemplo
&&	e (ambos verdadeiros?)	true && false (false)
 	ou (ao menos 1 verdadeiro?)	true false (true)
!	não (inverte V <-> F)	!true (false)

Exercício

Crie três variáveis (**a**, **b** e **c**). Atribua os valores 5, 8 e 3 a essas variáveis.

Verifique se **a** é igual a 6 e, ao mesmo tempo, **b** é menor ou igual a 5.

Atribua o resultado a **c**.

Apresente **c** em um parágrafo na página:

```
<script src="meu_script.js" defer></script>
</head>
<body>

    <p></p>

</body>
```


Resposta

```
let a = 5;  
let b = 8;  
let c = 3;  
  
c = (a == 6) && (b <= 5);  
  
document.querySelector("p").innerHTML = c;
```

Exercício 2

Repita o experimento utilizando como valores iniciais:

- $a = 6$
- $b = 5$
- $c = 300$

Outros operadores

Existem mais alguns outros operadores, como o ternário e os bitwise que não foram apresentados.

Assim, seguem algumas fontes para aqueles que desejarem se aprofundar:

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

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions and operators](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_operators)

Conditionais

Condicionais

Em JavaScript, os condicionais são similares (menos String) aos presentes em C e Java:

```
let cor = "Vermelha";
let versao = 7;

if (cor == "Vermelha" && versao > 5) {
    // Entra aqui se for a cor vermelha
    // e a versão maior que 5
} else if (cor == "Amarela") {
    // Entra aqui se não entrar no de cima
    // e a cor for amarela
} else {
    // Entra aqui caso não entre nos outros
}
```

Condicionais

Principais de condicionais: **IF** e **SWITCH**

```
let cor = "Vermelha";
let versao = 7;

if (cor == "Vermelha" && versao > 5) {
    // Entra aqui se for a cor vermelha
    // e a versão maior que 5
} else if (cor == "Amarela") {
    // Entra aqui se não entrar no de cima
    // e a cor for amarela
} else {
    // Entra aqui caso não entre nos outros
}
```

```
let animal = "Gato";

switch(animal) {
    case "Cachorro":
        console.log("É um cachorro");
        break;

    case "Gato":
        console.log("É um gato");
        break;

    case "Rato":
        console.log("É um rato");
        break;

    case "Pombo":
        console.log("É um pombo");
        break;
}
```

Exercício - IF

Crie duas variáveis, **numero** e **palavra**, crie um código que mostre ao usuário:

- **Cão** → se **numero** menor que 9 e **palavra** diferente de **coelho**
- **Gato** → se **numero** menor que 9 e **palavra** igual a **coelho**
- **Urso** → se não for nenhum dos outros e a **palavra** for igual a **lebre**

```
<script src="meu_script.js" defer></script>
</head>
<body>

    <p></p>

</body>
```

Exercício - SWITCH

Crie a variável, **numero** e mostre em um parágrafo:

- **Cão** → se o **numero** é igual a 3
- **Gato** → se o **numero** é igual a 7
- **Urso** → caso contrário

```
<script src="meu_script.js" defer></script>
</head>
<body>

    <p></p>

</body>
```


Listas (Arrays)

Lista (Array)

Uma lista é uma variável que pode acumular mais de um valor:

0	1	2	3
9	"Olá"	42	87

```
let lista = [9, "Olá", 42, 87]
```

Principais métodos:

- **lista.length**
 - Tamanho da lista
- **lista.push(item)**
 - Adiciona o **item**
- **lista.pop**
 - Remove último item
- **lista[n]**
 - Acessa o item **n**

Exercício

- Crie uma lista
- Adicione os itens: “Casa”, “Moto”, 900, 275, null, “Avião”
- Mostre os itens da lista no console
- Remova todos os itens da lista

Laços

Laços

Existem diversos tipos de laços em JS, alguns com aplicações bem específicas.

Os dois principais são:

```
for (let i=0; i < 100; ++i) {  
  console.log("Olá número " + i);  
}
```

```
let i = 0;  
while(i < 100) {  
  console.log("Olá número " + i);  
  
  i = i + 1;  
}
```

Exercício

Crie um laço que imprime no console apenas os números múltiplos de 4 entre 0 e 500.

```
for (let i=0; i < 100; ++i) {  
    console.log("Olá número " + i);  
}
```

```
let i = 0;  
while(i < 100) {  
    console.log("Olá número " + i);  
  
    i = i + 1;  
}
```

Funções

Funções

Uma função em JavaScript tem a seguinte sintaxe:

```
function mostrar(a, b, c) {  
    return a + b + c;  
}  
  
let a = 6;  
let resposta = mostrar(2, 5, a)
```


Funções

Uma função em JavaScript tem a seguinte sintaxe:

Define uma função

Nome

```
function mostrar(a, b, c) {  
    return a + b + c;  
}  
  
let a = 6;  
let resposta = mostrar(2, 5, a)
```

Parâmetros

Argumentos

Exercício

Crie as seguintes funções correspondendo a uma calculadora:

- somar (a, b)
- subtrair(a, b)
- multiplicar(a, b)



https://img.kalunga.com.br/fotosdeprodutos/159008z_1.jpg

Arrow functions

Podemos simplificar as funções utilizando a sintaxe de Arrow functions:

```
let mostrar = (a, b, c) => { return a*b*c; };
```

```
let mostrar = (a, b, c) => a*b*c;
```

- Tiramos a palavra **function**
- Adicionamos **=>** entre a assinatura e o corpo da função
- Se for apenas um retorno, podemos tirar as **chaves** e o **return**
- São úteis ao passarmos funções por parâmetro para outras funções (ex. Express)

Exercício

Crie as seguintes funções correspondendo a uma calculadora (como Arrow Functions):

- somar (a, b)
- subtrair(a, b)
- multiplicar(a, b)

```
let mostrar = (a, b, c) => { return a*b*c; };
```

```
let mostrar = (a, b, c) => a*b*c;
```



https://img.kalunga.com.br/fotosdeprodutos/159008z_1.jpg

Interações com uma página WEB

Seletores

Seletores

Existem diversos meios de selecionar elementos em uma página:

- Podemos selecionar pelo id

```
document.getElementById("meuId");
```

- Podemos selecionar pela classe

```
document.getElementsByClassName("minha-classe");
```

- Podemos utilizar os seletores do CSS

```
document.querySelectorAll("body>p.minha-classe");
```

- Podemos pegar apenas o primeiro elemento que corresponde ao seletor CSS

```
document.querySelector("body>p.minha-classe");
```

Seletores

Existem diversos meios de selecionar elementos em uma página:

- Podemos selecionar pelo id

```
document.getElementById("meuId");
```

- Podemos selecionar pela classe

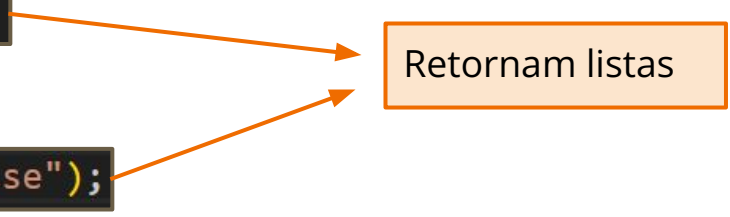
```
document.getElementsByClassName("minha-classe");
```

- Podemos utilizar os seletores do CSS

```
document.querySelectorAll("body>p.minha-classe");
```

- Podemos pegar apenas o primeiro elemento que corresponde ao seletor CSS

```
document.querySelector("body>p.minha-classe");
```



Retornam listas

Exercício (1)

Dada a seguinte página WEB:

- Utilize o seletor de id para obter o parágrafo com id="meuid"
- Utilize o seletor de classe para obter os parágrafos com classe "vermelho"
- Utilize o querySelectorAll para selecionar todos os parágrafos pares
- Utilize o querySelector para selecionar o primeiro parágrafo dentro da div

```
<p>Primeiro</p>
<p id="meuid" class="vermelho">Segundo</p>
<p class="vermelho">Terceiro</p>
<div>
  <p>Primeiro div</p>
  <p class="vermelho">Segundo div</p>
</div>
```

Exercício (2)

Dada a seguinte página WEB:

- Altere a cor de cada elemento selecionado para vermelho.
 - Aplique apenas um seletor por vez para ver o resultado
 - Utilize comentários para desativar os demais

```
<p>Primeiro</p>
<p id="meuId" class="vermelho">Segundo</p>
<p class="vermelho">Terceiro</p>
<div>
  <p>Primeiro div</p>
  <p class="vermelho">Segundo div</p>
</div>
```

Eventos

Execução do código

Um código JavaScript pode ser executado em uma página WEB de duas maneiras:

- Logo que é inserido na página
 - Código fora de funções
- Quando ocorre um evento
 - Clique
 - Carregamento da página
 - Aperto de botão
 - Etc...



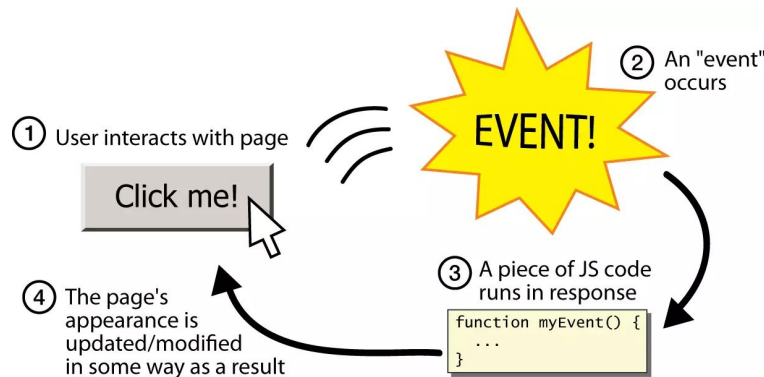
Eventos

Eventos são coisas que acontecem com nossa página

- Clique em elemento
- Perda de foco (clicar fora de um formulário)
- Página carregada
- ...

Lista de eventos:

https://www.w3schools.com/jsref/dom_obj_event.asp



Eventos

Podemos determinar a função em JavaScript para tratar cada evento pelo HTML:

```
<div onclick="mudar_cor()"></div>
```



```
function mudar_cor() {  
    let minha_div = document.querySelector("div");  
  
    minha_div.style.color = "red";  
}
```

Exercício

Crie funções em JavaScript para alterar a cor de cada um dos parágrafos da página para vermelho, vinculando com o elemento correspondente:

```
<body>

  <p>Olá</p>

  <p>tudo bem</p>

  <p>Com você?</p>

</body>
```

this

Podemos simplificar nosso código utilizando uma única função, com a palavra chave **this**


```
<p onclick="cor(this)">Olá</p>  
  
<p onclick="cor(this)">tudo bem</p>  
  
<p onclick="cor(this)">Com você?</p>
```

```
function mudar_cor(paragrafo) {  
    paragrafo.style.color = "red";  
}
```


this

Podemos simplificar nosso código utilizando uma única função, com a palavra chave **this**

```
<p onclick="cor(this)">Olá</p>  
<p onclick="cor(this)">tudo bem</p>  
<p onclick="cor(this)">Com você?</p>
```



```
function mudar_cor(paragrafo) {  
    paragrafo.style.color = "red";  
}
```

Exercício

```
function mudar_cor(paragrafo) {  
    paragrafo.style.color = "red";  
}
```

```
<p onclick="cor(this)">Olá</p>
```

Crie uma função em JavaScript que altere a cor de fundo dos elementos da página para vermelho, ao serem clicados:

```
<body>  
  
    <p>Olá</p>  
  
    <p>tudo bem</p>  
  
    <div></div>  
  
    <p>Com você?</p>  
  
    <div></div>  
  
</body>
```

Eventos interessantes

Dentre os diversos eventos existentes, alguns se destacam:

- onclick
 - Elemento recebe um clique
- onmouseenter
 - Mouse passa sobre o elemento
- onmouseleave
 - Mouse sai de cima do elemento

Alterando atributos

Alterando atributos

Utilizando JavaScript podemos alterar qualquer atributo de um elemento HTML:

```
let elemento = document.getElementById('id');  
elemento.setAttribute("atributo", "valor");
```

```
elemento.setAttribute("src", "praia.png");  
elemento.setAttribute("style", "width: 350px;");
```

Exercício

```
elemento.setAttribute("atributo", "valor");
```

- onclick
- onmouseenter
- onmouseleave

Modifique a página utilizando JavaScript para que:

- Imagem **comece** com um **esquilo**
- Ao **clicar** no parágrafo, imagem troque para uma **raposa**
- Ao **clicar** novamente, ela volte para o **esquilo**
- Se o mouse estiver **sobre** o parágrafo: imagem seja alterada para uma **Harpia**
- Se o mouse **sair**, imagem deve voltar à **anterior**

```
<img src="" alt="">  
  
<p>Trocar imagem</p>
```

Vários eventos

Vinculando eventos

Em determinadas situações, podemos precisar vincular diversos eventos a um único elemento ou não teremos como alterar o HTML da página

Assim, podemos utilizar o próprio JavaScript para adicionar eventos!

```
let elemento = ...;  
elemento.addEventListener("evento", Código JS);
```

```
elemento.addEventListener("click", mudar_cor());
```



```
elemento.setAttribute("atributo", "valor");
```

Exercício

```
elemento.addEventListener("click", mudar_cor());
```

- onclick
- onmouseenter
- onmouseleave

Repita o exercício anterior, porém sem editar o HTML

- Imagem **comece** com um **esquilo**
- Ao **clicar** no parágrafo, imagem troque para uma **raposa**
- Ao **clicar** novamente, ela volte para o **esquilo**
- Se o mouse estiver **sobre** o parágrafo: imagem seja alterada para uma **Harpia**
- Se o mouse **sair**, imagem deve voltar à **anterior**

```
<img src="" alt="">  
  
<p>Trocar imagem</p>
```

Material complementar

Como o assunto é demasiadamente grande para nosso total de aulas, sugiro a leitura dos seguintes materiais:

<https://www.w3schools.com/js/default.asp>

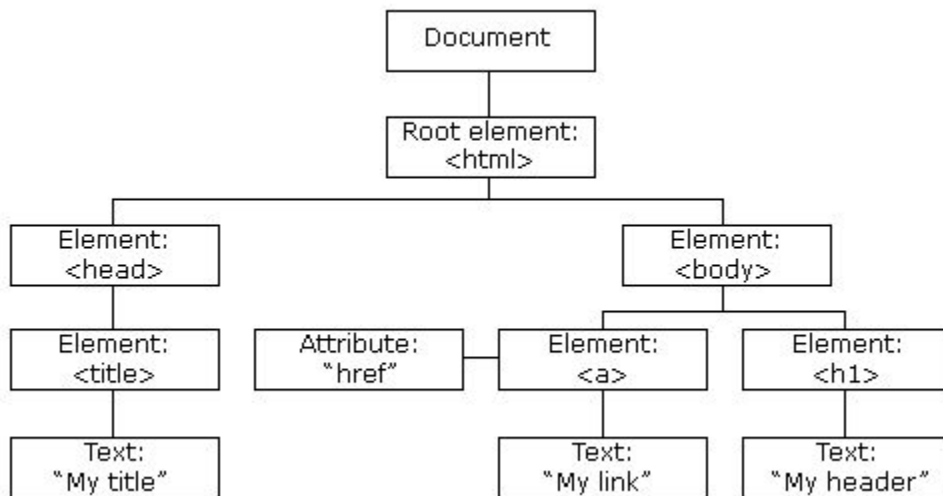
<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>

DOM

Document Object Model

Document Object Model

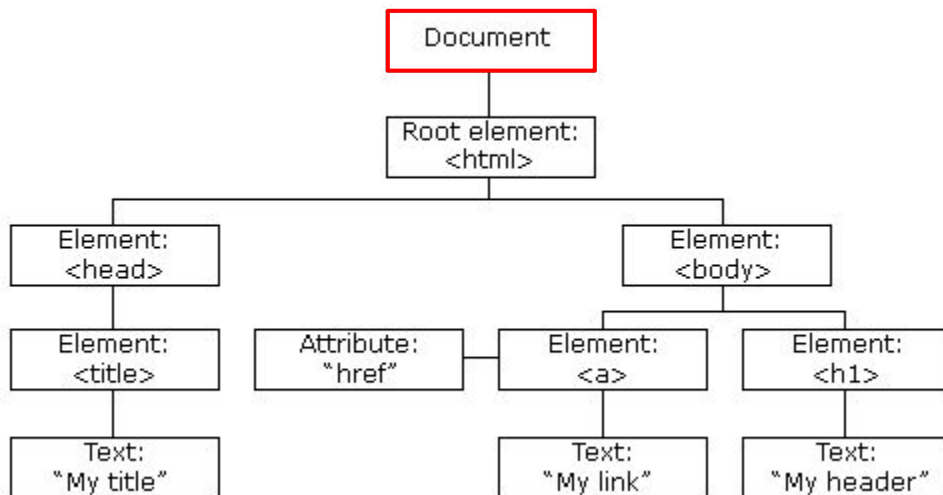
Quando uma página é carregada, é construída uma árvore contendo todos os seus elementos:



https://www.w3schools.com/js/pic_htmltree.gif

Document Object Model

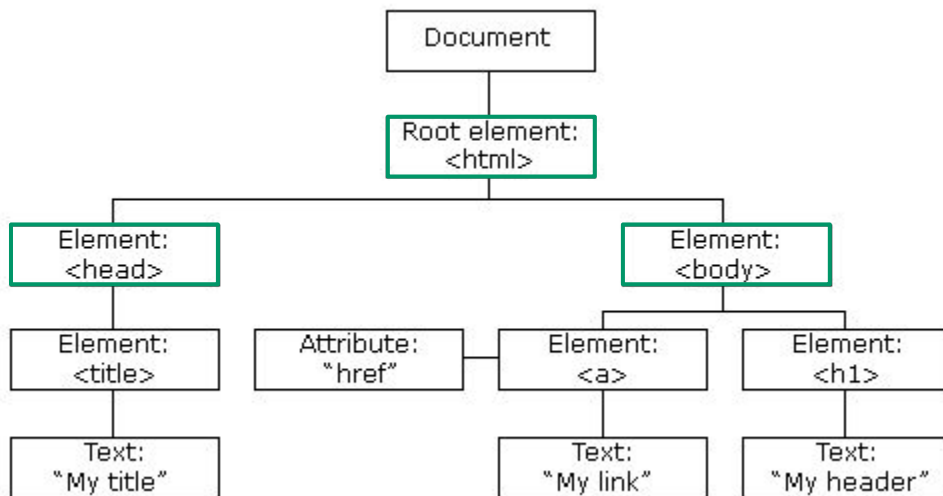
Quando uma página é carregada, é construída uma árvore contendo todos os seus elementos:



https://www.w3schools.com/js/pic_htmltree.gif

Document Object Model

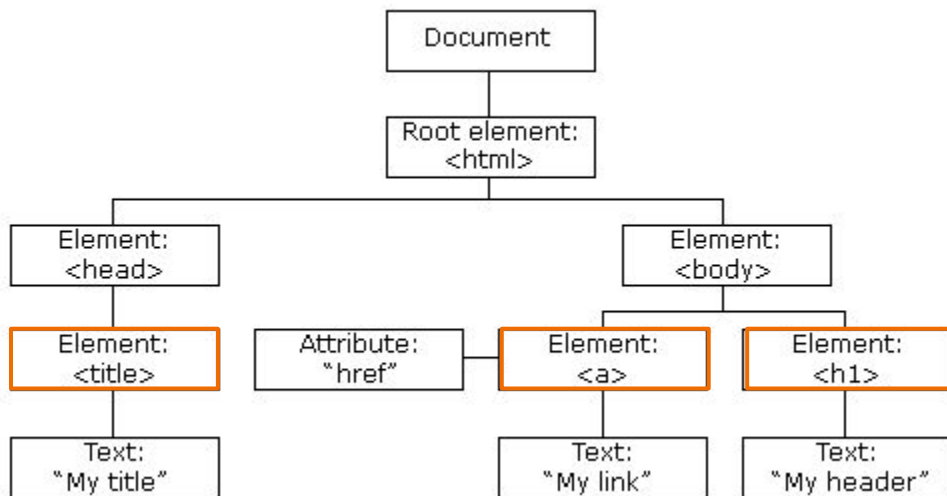
Quando uma página é carregada, é construída uma árvore contendo todos os seus elementos:



https://www.w3schools.com/js/pic_htmltree.gif

Document Object Model

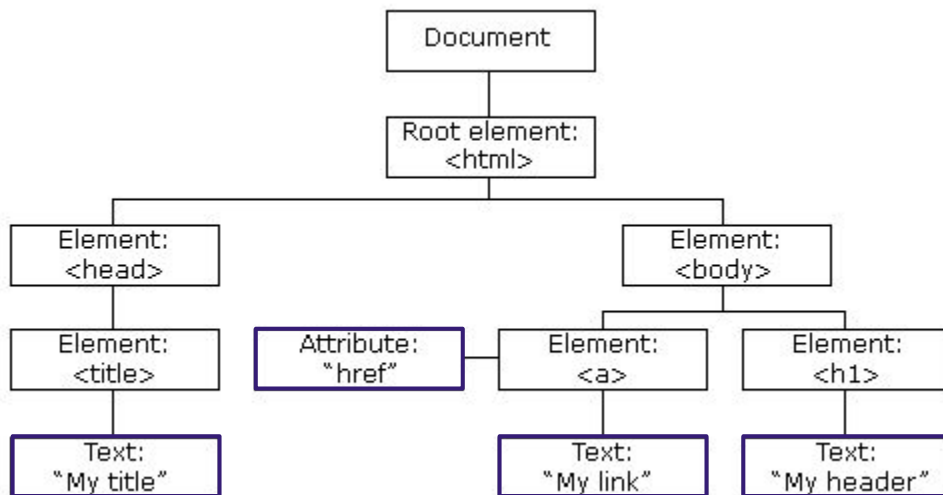
Quando uma página é carregada, é construída uma árvore contendo todos os seus elementos:



https://www.w3schools.com/js/pic_htmltree.gif

Document Object Model

Quando uma página é carregada, é construída uma árvore contendo todos os seus elementos:

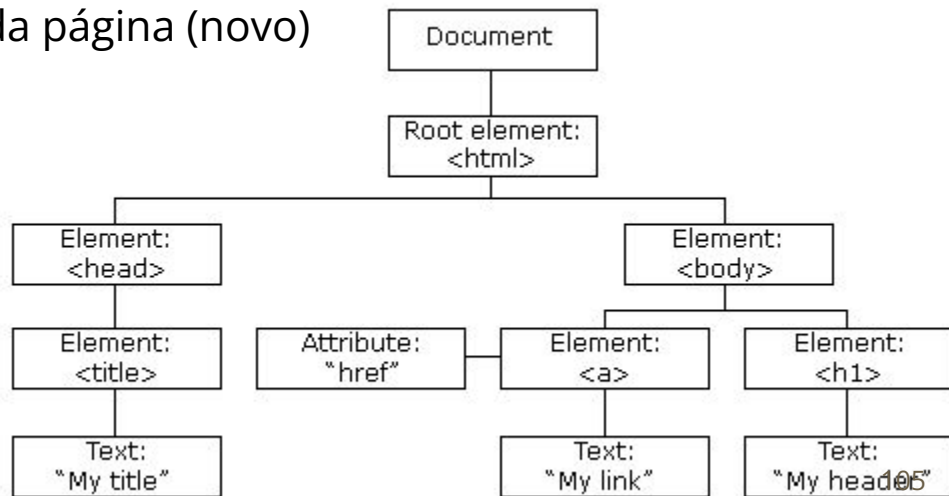


https://www.w3schools.com/js/pic_htmltree.gif

DOM

Tendo acesso ao DOM, podemos:

- alterar qualquer valor/atributo (como visto anteriormente)
- reagir a eventos sobre elementos (como visto anteriormente)
- Adicionar/alterar/remover elementos da página (novo)



Adicionando elementos

Para adicionar um elemento a uma página, precisamos:

1. Criar o elemento:

```
const elemento = document.createElement("TAG");
```

2. Adicionar ele à página:

- a. Seleccionamos o pai dele com os seletores já apresentados
- b. Adicionamos o elemento criado como filho do pai selecionado

```
pai.appendChild(elemento);
```

- c. Por fim, podemos editar seu conteúdo e atributos com o que aprendemos anteriormente!

Exemplo

Dada a seguinte página WEB, vamos adicionar um parágrafo contendo “Hello World JS” escrito em vermelho dentro da DIV.

```
<body>  
  
    <div></div>  
  
</body>
```

Removendo elementos

Navegadores mais novos permitem remover um elemento utilizando o método **remove()**

```
const p = document.querySelector("p");  
  
p.remove();
```

Para navegadores mais antigos, precisamos remover o elemento por meio de seu pai:

```
const p = document.querySelector("p");  
  
p.parentNode.removeChild(p);
```

```
const elemento = document.createElement("TAG");
```

Exercício

```
pai.appendChild(elemento);
```

```
p.parentNode.removeChild(p);
```

Dada a seguinte página, remova a imagem e a substitua por um parágrafo com o texto “Aqui havia uma imagem”.

```
<body>  
  
    <p>Bem vindos!</p>  
    <img src="" alt="">  
  
</body>
```

Exercício

Dada a seguinte página:

```
<body>  
  <p>Clique para ver o peixe!</p>  
</body>
```

Quando o parágrafo é clicado, substitua o parágrafo por uma imagem de um peixe.

Quando a imagem é clicada, ela deve ser removida e o parágrafo voltar.

Esse ciclo deve se repetir indefinidamente.

Dica: Uma função vinculada ao texto e uma à imagem.