

# FrontEnd

## DOM

## Document Object Model

Conjunto de objetos que são interpretados quando manipulamos uma aplicação web.

Ela se baseia em uma árvore.

# FrontEnd - JavaScript

## DOM

## Document Object Model

Conseguimos alterar diversos elementos HTML e CSS usando a programação JavaScript

Um dos grandes responsáveis por isso tudo é o objeto `document` que é responsável por conceder ao código Javascript todo o acesso a árvore DOM do navegador Web.

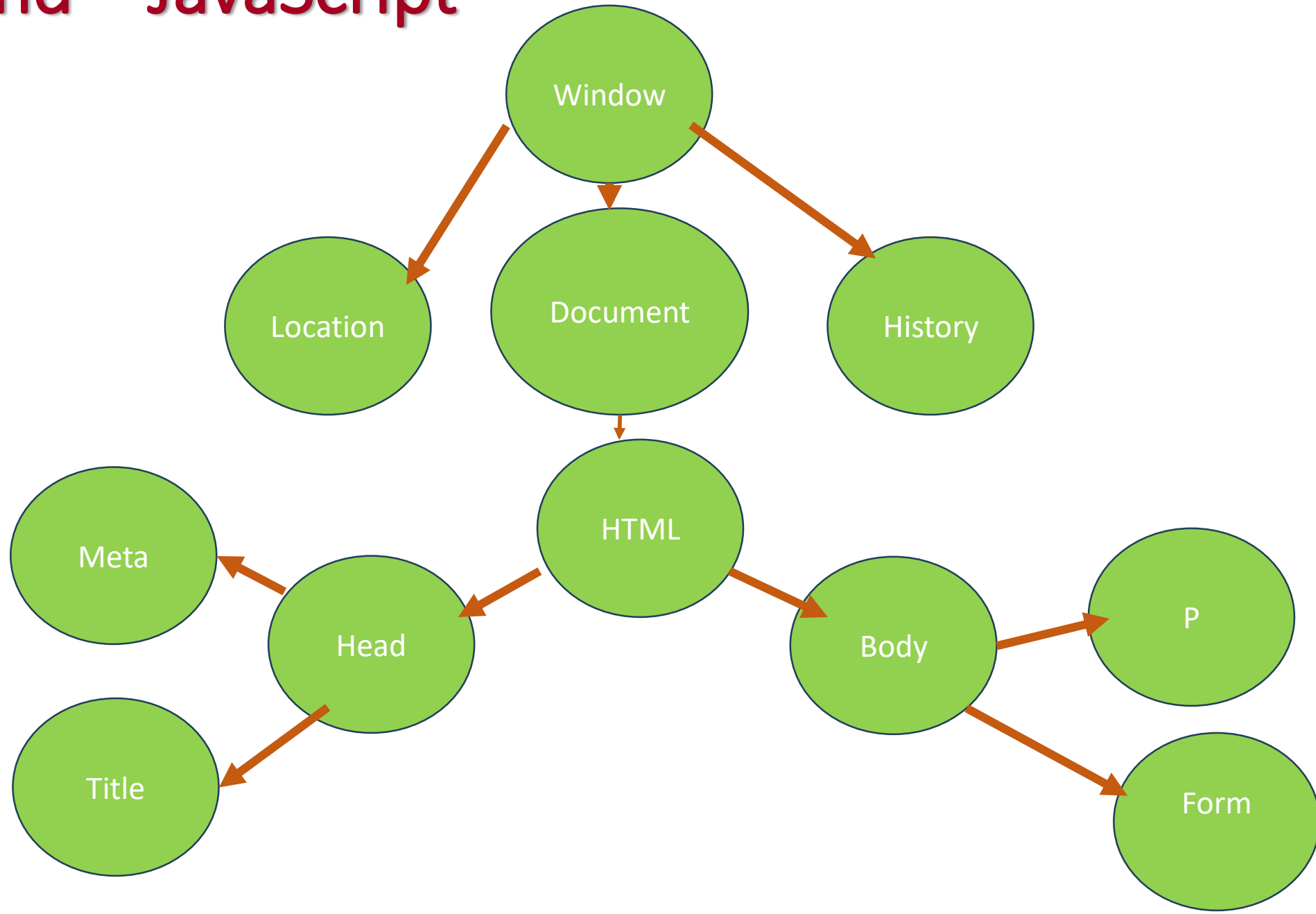
Portanto, qualquer coisa criada pelo navegador Web no modelo da página Web poderá ser acessado através do objeto Javascript `document`.

# FrontEnd - JavaScript

## DOM -Document Object Model

- Window = Janela. O que vai acontecer na janela do navegador.
- Ele tem filhos como: Document, location, history
- O document, tem o filho HTML, por exemplo  
Que tem os filhos: head e body
- Onde no Head tem o meta e Title, e o body tem h1, p, div forms.....

# FrontEnd - JavaScript



# FrontEnd - JavaScript

```
Front_aula04.html > html > body > script > h1
1  <html>
2    <head>
3      <title>
4        Minha Aula04
5      </title>
6
7    </head>
8    <body>
9      <script>
10     var h1 = window.document.getE
11
12     </script>
13     <h1>Minha Aula04</h1>
14
15   </body>
16
17 </html>
```

- getElementB... (method) Document.getElementById(elementI...
- getElementsByClassName
- getElementsByName
- getElementsByTagName
- getElementsByTagNameNS
- getSelection
- getRootNode
- ☐ s-get-content svelte-get-context
- ☐ s-store-get-value svelte-store-get

Quando começamos a digitar a instrução, o editor já consegue nos ajudar, vendo quais são os “filhos” das instruções que já inseridas.

# FrontEnd - JavaScript

```
01_aula04.html > html > body > script
<html>
<head>
  <title>Minha Aula04</title>
</head>
<body>
  <h1>Minha Aula04</h1>

  <script>
    var h1 = window.document.getElementsByTagName('h1')[0];
    var corpo = window.document.body
    h1.style.color = 'white';
    corpo.style.background = 'blue';
  </script>
</body>
</html>
```

Podemos alterar as características do componente da tela usando o `getElementsByTagName` por exemplo.

Colocamos a tag que sejam e em qual posição que desejamos fazer a alteração. No exemplo tem a posição `[0]`, ou sejam na primeira vez em que a tag `<h1>` é chamada.

# FrontEnd - JavaScript

```
corpo.style.background = 'blue'; // coloca o fundo azul  
  
//pagar um elemento do HTML pelo ID  
var obs = window.document.getElementById('obs')// pega o elemento com o id OBS  
obs.style.background = 'purple'; // coloca o fundo roxo  
obs.innerText = 'Obs, eu sou uma Div com id'; //muda o texto pela programação
```

Podemos alterar as características do componente da tela usando o `getElementbyID` também.

Resgatamos o item do HTML que tem o Id solicitado (obs), e fazemos alguma ação com ele, por exemplo colocar em roxo, e alteramos o texto exibido.



# FrontEnd - JavaScript

```
// Pegar o elemento pelo querySelector  
var b = window.document.querySelector('div#selector');  
b.style.background = 'red'; // Coloca o fundo vermelho  
b.innerText = "Olá, eu sou o selector"; // Define o texto
```

Podemos alterar as características do componente da tela usando o querySelector.

Resgatamos o item do HTML que tem o Id solicitado (div#selector), tag e id, e podemos manipular a apresentação da forma que preferirmos.

# FrontEnd - JavaScript

```
<body>
  <h2 id="exemplos">Integração do JS com HTML e CSS</h2>

  <fieldset class="container-exemplos">
    <div class="container-mensagem">
      <h2 id="exemplo-lista">Lista</h2>

      <label for="nome" class="item label">Nome</label>
      <input id="nome" class="item campo" type="text" />

      <button id="btn-mensagem" class="item botao">Enviar</button>

      <label for="mensagem" class="item label">Mensagem</label>
      <p id="mensagem" class="item saida"></p>

      <ol id="lista" class="item"></ol>
    </div>
  </fieldset>
</body>
```

# FrontEnd - JavaScript

```
function listar() {  
    // Apresenta a mensagem  
    mensagem.innerHTML = "Olá! " + nome.value  
    // Cria um elemento HTML li e atribui na variável item  
    let item = document.createElement("li")  
    // Acrescenta o nome digitado no item  
    item.innerHTML = nome.value  
    // Insere o novo elemento item (li) como filho do elemento lista (ol)  
    // já existente no documento HTML  
    lista.appendChild(item)  
    // Limpa o input  
    nome.value = ""  
    // E posiciona o curso para a próxima inserção  
    nome.focus()  
}
```

# FrontEnd - JavaScript

---

## Integração do JS com HTML e CSS

---

### Lista

Nome

Mensagem

---

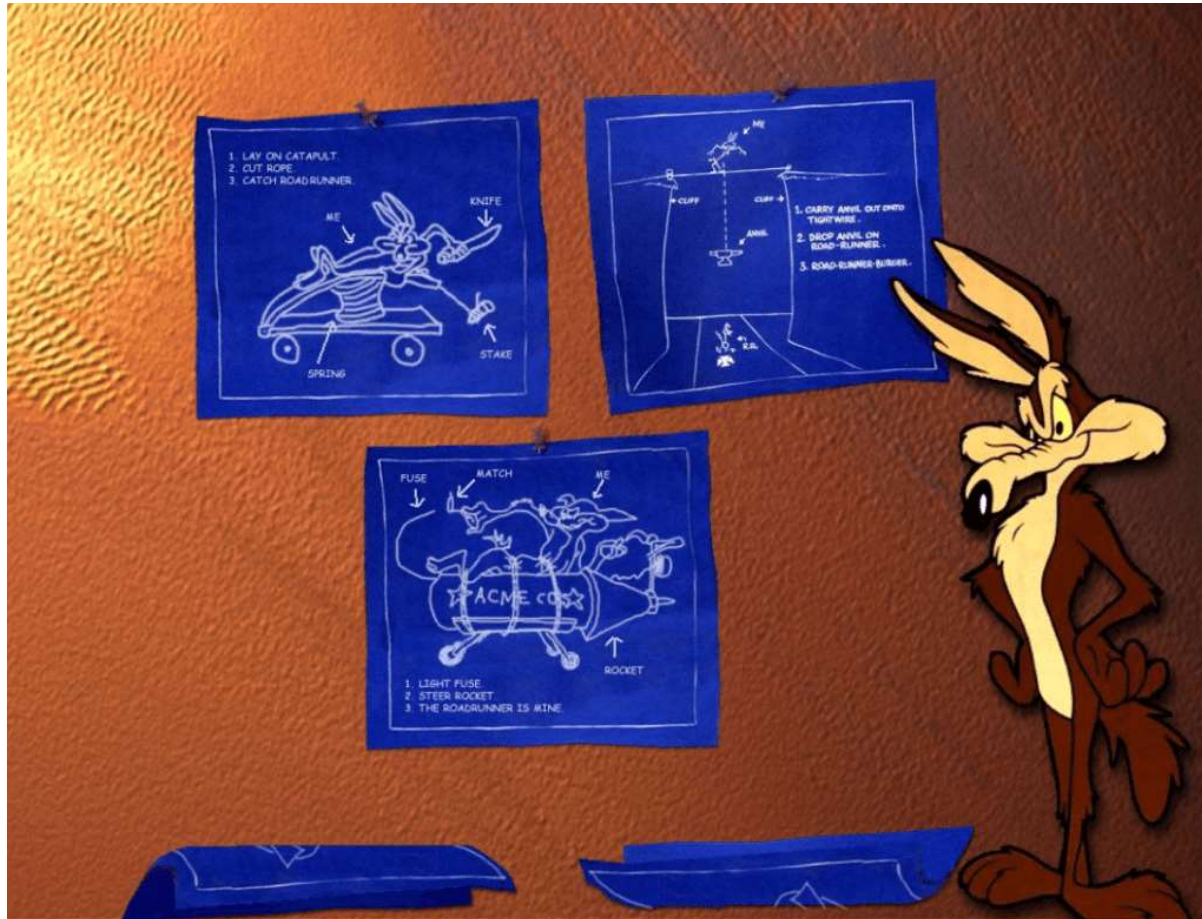
## Função

Funções são blocos de construção fundamentais em Javascript.

Elas permitem que conseguimos executar uma série de comandos a fim de fazer o processamento de alguma ação que desejemos, e geremos um resultado.



## Função



Habitualmente esses blocos são planejados, etapa por etapa, nomeados e chamados nos momentos mais oportunos.

## Função

Para usar uma função, devemos fazer a declaração dela, com a seguinte sintaxe.



The diagram illustrates the syntax for declaring a function in JavaScript. It shows the code `function name() {}` with four labels and arrows pointing to specific parts:   
- **palavra reservada** (reserved word) points to `function`.   
- **nome da função** (function name) points to `name`.   
- **parenteses** (parentheses) points to the opening parenthesis `(`.   
- **chaves** (braces) points to the closing curly brace `}`.

```
function name() {}
```

## Função

Por exemplo, se eu quisesse uma função para calcular um número ao quadrado.

```
function square(numero) {  
    return numero * numero;  
}
```



## Função

Nas funções posso fazer cálculos aritméticos;  
Posso ter expressões condicionais;  
Posso ter laços de repetição;  
Posso ter outras funções;  
Tratar dados recebidos pelo usuário;  
Processar interação com o usuário.

## Função

```
function somarVarios(n1, n2, n3 = 0, n4 = 10){  
    soma = n1 + n2 + n3 + n4  
    return soma  
}  
  
function verificarpositivo(n1,n2,n3,n4){  
    if((n1<0) || (n2<0) || (n3<0) || (n4<0)){  
        negativo = "há pelo um numero negativo"  
        return negativo  
    }  
    else{  
        negativo = "todos são positivos"  
        return negativo  
    }  
}
```

# Função Anônima

São funções que não recebem nomes, que são executadas no momento da execução, e que não será chamada em outra parte do aplicativo.

Normalmente são declaradas junto à uma variável.

# Função Anônima

```
const subtrair = function(n1, n2){  
    return n1 - n2  
}
```

```
window.alert(subtrair(40,30))
```

## Parâmetros

```
function somarVarios(n1, n2, n3 , n4 ){  
    return n1 + n2 + n3 + n4  
}
```

```
alert( somarVarios(2, 4, 6, 8, 10) ) 20  
alert( somarVarios(2, 4, 6, 8) ) 20  
alert (somarVarios(2, 4, 6) ) NAN  
alert( somarVarios(2, 4) ) NAN
```

## Parâmetros

```
function somarVarios(n1, n2, n3 , n4 ){  
    return n1 + n2 + n3 + n4  
}
```

```
alert( somarVarios(2, 4, 6, 8, 10) ) 20  
alert( somarVarios(2, 4, 6, 8) ) 20  
alert (somarVarios(2, 4, 6) ) NAN  
alert( somarVarios(2, 4) ) NAN
```

# Parâmetros

```
function somarVarios(n1 =0, n2=0, n3=0 , n4=0 ){  
    return n1 + n2 + n3 + n4  
}
```

```
alert( somarVarios(2, 4, 6, 8, 10) )  
alert( somarVarios(2, 4, 6, 8) )  
alert (somarVarios(2, 4, 6) )  
alert( somarVarios(2, 4) )
```

# Arrow Function

Em termos simples, uma arrow function é uma forma concisa de escrever uma função em JavaScript. Ela otimiza a escrita do seu código, deixando-o mais limpo, enxuto e aumentando a legibilidade.



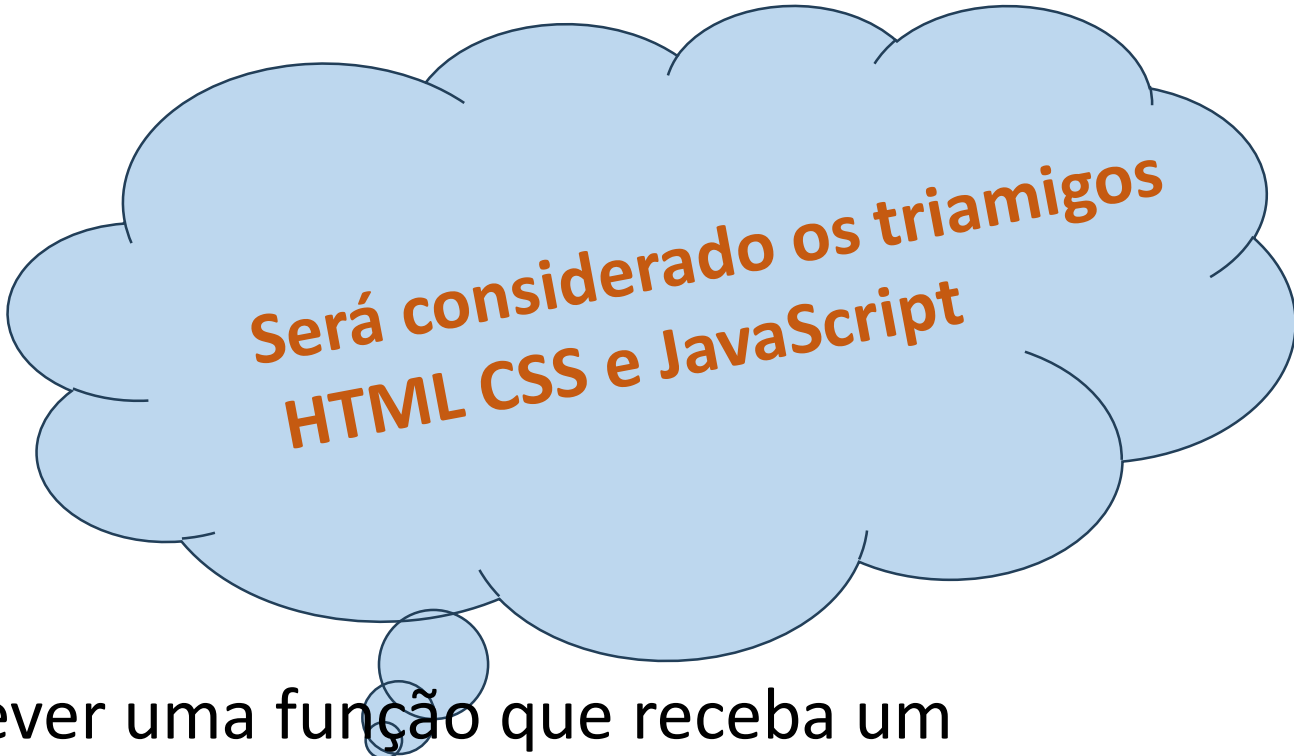
## Arrow function

```
const dividirArrowComCorpo = (n1, n2) => {  
  return n1 / n2  
}
```

Se eu tiver só uma linha de execução diminuo ainda mais as expressões

```
// Arrow function (com uma linha de instrução)  
const dividirArrow = (n1, n2) => n1 / n2
```

## Exercícios



Será considerado os triamigos  
HTML CSS e JavaScript

Neste exercício, você precisa escrever uma função que receba um número como argumento e retorne o quadrado desse número (ou seja, o número multiplicado por ele mesmo).

Neste exercício, você precisa escrever uma função que receba um número como argumento e retorne uma string indicando se o número é par ou ímpar.

## Exercícios

Neste exercício, você precisa escrever uma função que receba três números como argumentos e retorne o maior desses números.

Faça uma calculadora, onde o usuário possa informar 2 valores, e escolher a operação desejada: Soma, subtração, multiplicação, divisão, exponenciação, raiz quadrada, percentual, Fibonacci e fatorial, e exiba o resultado ao usuário.

## Exercícios

Faça o formulário para o cadastro de um cliente novo para uma livraria com:

Nome (validar se o campo não está vazio)

E-mail: (validar se tem “@” e “.”)

Endereço: Validar se está preenchido

Telefone: (validar se está preenchido (desafio, ver se só tem números)

CPF: (validar se está preenchido (desafio, ver se só tem números).

No caso de algum campo entrar em divergência quando selecionado o botão “Enviar”, mostrar um alert ao usuário, e contornar o campo em vermelho.

Todos os campos são obrigatórios!

## Exercícios

Faça o formulário para que um aluno possa visualizar seu status em um curso, onde deve:

Colocar 3 nota, com seus devidos pesos. Por exemplo:

$$\frac{(\text{nota1} \times \text{peso1}) + (\text{nota2} \times \text{peso2}) + (\text{nota3} \times \text{peso3})}{(\text{peso1} + \text{peso2} + \text{peso3})}$$

Exibir se o aluno foi

- Aprovado com excelência (notas  $\geq 9$ ),
- Aprovado (nota  $\leq 6$ ) e
- “Reprovado” nos demais casos.

No caso de reprovado, deve ser possível inserir uma 4 nota valendo um peso de 40%, para ver se consegue a sua aprovação.

# FrontEnd - JavaScript

# Exercícios

Enviar para:

Fernanda.fretes@sp.senai.br