



# INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

AULA 08 - EVENTOS DOM

# O QUE IREMOS APRENDER

**01** RESUMO DA AULA PASSADA

**02** O QUE É DOM

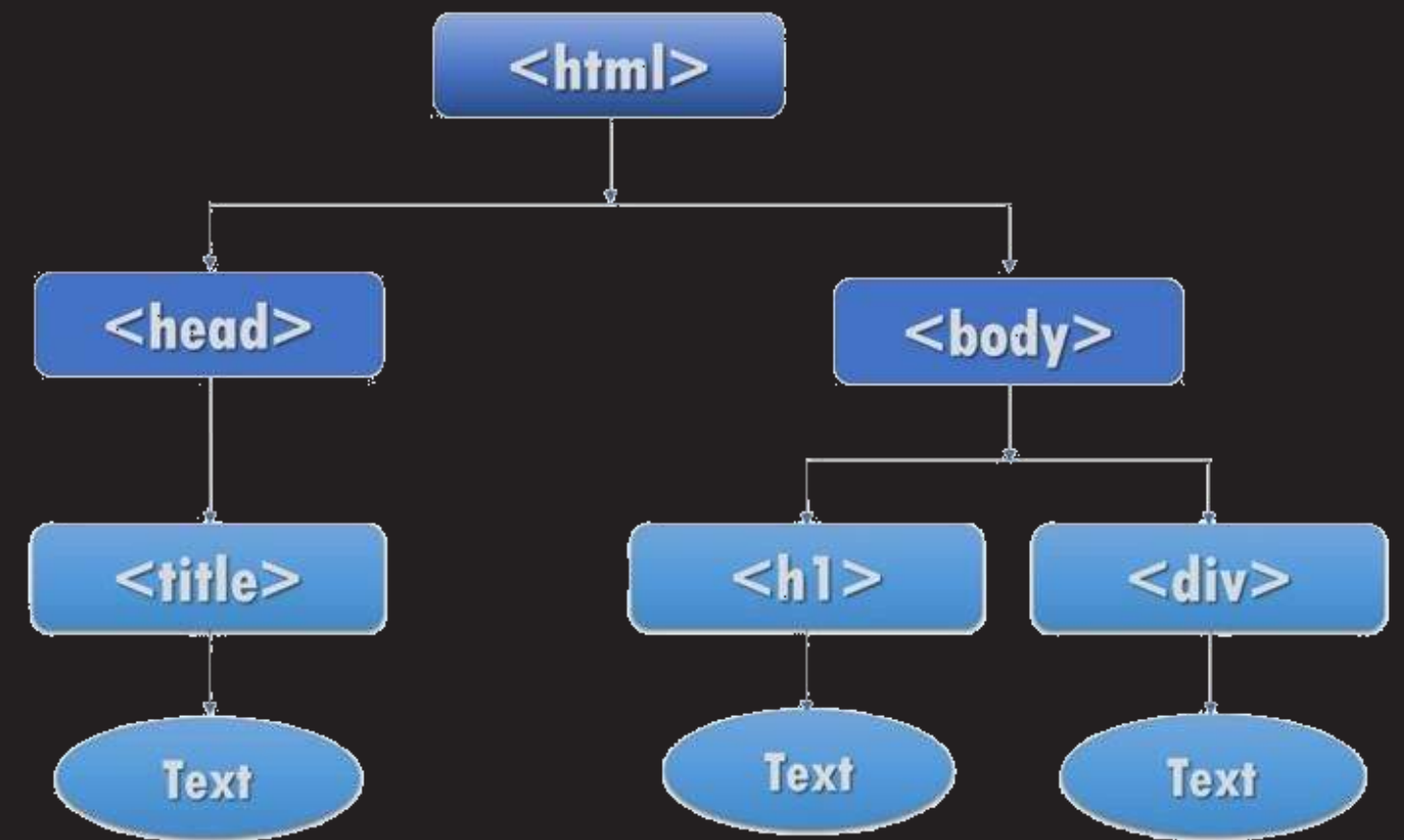
**03** DOM

**04** MÃOS NO CÓDIGO

# O QUE É DOM

DOM (Document Object Model) é uma representação da nossa árvore de elementos do HTML. Ela nos traz todo o modelo do nosso documento HTML, nos permitindo manipulá-lo de diversas formas.

Ela se assemelha muito a uma árvore genealógica, onde cada elemento tem seus respectivos pai e filhos. No entanto, cada elemento pode ter apenas um elemento pai, mas diversos elementos filhos.



# DOM

A primeira coisa que vamos aprender a fazer é a pegar elementos da DOM, ou seja, pegar elementos do nosso documento HTML, para que possamos manipulá-los.

Primeiro precisamos pegar nosso documento, temos acesso a ele através da variável global “document”, depois utilizamos um dos métodos disponíveis.

Para isso, temos diversos meios, sendo os mais básicos:

`getElementsByTagName`      `getElementsByTagName`

`getElementsByTagName`      `getElementById`

# DOM

O `getElementsByTagName` nos permite pegar todos os elementos com a mesma tag.

No exemplo a seguir, pegamos todos os elementos que são da tag p.




```
1  <!-- Definindo a tag p no HTML -->
2  <p></p>
3
4  <script>
5    // Introduzindo a tag P no JavaScript a partir da tag
6    let paragrafo = document.getElementsByTagName('p');
7
8  </script>
9
```



# DOM

O `getElementsByName` nos traz todos os elementos que possuírem um atributo name específico.

No exemplo a seguir, pegamos todos os elementos que possuírem o atributo name com o valor **nomeQualquer**.



```
1  <!-- Definindo a tag <p> no html -->
2  <p name="nomeQualquer"></p>
3  <p name="nomeQualquer"></p>
4  <script>
5      //Introduzindo a tag <p> a partir do name = 'nomeQualquer'
6      let paragrafo = document.getElementsByName('nomeQualquer');
7      // Escrevendo no html na Primeira tag
8      paragrafo[0].innerText = ' test'
9      // Escrevendo no html na Segunda tag
10     paragrafo[1].innerText = ' test2'
11 </script>
```



# DOM

O `getElementsByClassName` vai nos trazer todos os elementos com uma mesma classe.

No exemplo a seguir, o seletor vai nos trazer todos os elementos que possuírem uma classe **minhaClasse**.



```
1  <!-- Definindo a tag <p> no html -->
2  <p class="minhaClass"></p>
3
4  <script>
5      //Introduzindo a tag <p> a partir da class = 'minhaClass'
6      let paragrafo = document.getElementsByClassName('minhaClass');
7      // Escrevendo no html na tag <p>
8      paragrafo[0].innerText = 'escreveno no html'
9  </script>
10
```

# DOM

O `getElementById` vai nos trazer o primeiro elemento que ele encontrar com o id que passarmos para ele.

No exemplo a seguir, vai nos trazer o primeiro elemento que possuir o id **texto**.

```
1
2  <!-- Definindo a tag p no HTML com o id "texto" -->
3  <p id="texto"></p>
4  <script>
5      // Introduzindo todos elementos com o id "texto" na variavel paragrafos
6      let paragrafos = document.getElementById('texto');
7  </script>
```





# DOM

Além dos métodos anteriores, nós vamos ter um meio um pouco mais avançado e moderno. Eles são o `querySelector` e o `querySelectorAll`, eles vão pegar os elementos através de um seletor, como no CSS. Além de nos trazer um elemento em um formato com várias funcionalidades novas.

Por exemplo:

- Com o `querySelector` podemos pegar elementos que estão dentro de outros elementos
- O formato que o `querySelectorAll` torna possível a utilização das funções agregadoras, enquanto com `getElementsBy*` nós só podemos percorrer os elementos através de um `for ... of`.



# DOM

Para utilizar o `querySelector` é simples, vamos pegar primeiro o documento, da mesma forma que os anteriores, e depois vamos passar o método `querySelector`, passando na chamada da função o seletor que você quer. Os seletores vão seguir o mesmo padrão dos do CSS.

No exemplo a seguir, vamos pegar o primeiro elemento do documento com a tag `p`.

```
1  <!-- Definindo somente a tag p -->
2  <h1></h1>
3  <!-- Definindo a tag p no HTML com o id = "paragrafo" -->
4  <p id="paragrafo"></p>
5  <!-- Definindo a tag p no HTML com a class = "paragrafo" -->
6  <p class="texto"></p>
7  <script>
8      // Introduzindo a tag P a partir da tag
9      let h1 = document.querySelector('h1');
10     // Introduzindo a tag P a partir do id
11     let paragrafo = document.querySelector('#paragrafo');
12     // Introduzindo a tag P a partir da classe
13     let texto = document.querySelector('.texto');
14 </script>
```



# DOM

O `querySelectorAll` é muito similar ao `querySelector`, porém ele vai nos trazer um array dos elementos que ele encontrar.

No exemplo a seguir, pegamos um array com todos os elementos com a classe `minhaClasse` que encontrar no documento.

```
1  <!-- definido as tags HTML com a classe 'minhaClasse' -->
2  <h1 class="minhaClasse"></h1>
3
4  <p class="minhaClasse"></p>
5
6  <script>
7      // introduzindo elementos com classe 'minhaclasse' na variavel paragrafos
8      let paragrafos = document.querySelectorAll('.minhaClasse')
9  </script>
```



# ATIVIDADE PRÁTICA

## **Atividade 01**

Crie um arquivo JavaScript e implemente a função `getValue` que irá pegar o valor do input e exibi-lo no console.

## **Atividade 02**

Crie um arquivo JavaScript e implemente a função `valorInput` que irá pegar o valor do input e coloque em um parágrafo o valor que foi resgatado do input.

# DOM

A DOM também nos permite criar novos elementos, nos dando a possibilidade de inserir novos elementos no nosso documento.

Fazemos isso através do `createElement`.

Para criar um novo elemento primeiro pegamos nosso document, através do `document`, em seguida chamamos a função `createElement` e passamos a ela a tag do elemento que queremos criar. No exemplo a seguir, criamos um elemento com a tag **div**.

# DOM

No exemplo a seguir, criamos um elemento com a tag **div**.



```
1  <script>
2  // Criando novo elemento no html a partir do "createElement"
3  let divNova = document.createElement("div")
4  </script>
```



# DOM

Agora que sabemos como pegar e criar elementos no nosso documento, podemos finalmente aprender a como manipular esses elementos. Podemos manipular os elementos de diversas formas:

## Através de métodos

- Adicionar elementos filhos
- Alterar atributos
- Manipular classes

## Diretamente

- Alterar texto interno
- Alterar estilo
- Alterar atributos
- Receber valores

# DOM

O método `appendChild` vai adicionar um elemento filho ao elemento. Utilizamos ele da seguinte forma:

```
1  <h1 id="meuElemento" class="minhaClasse">titulo</h1>
2
3  <p class="minhaClasse">paragrafo 1</p>
4
5  <script>
6
7      const meuElemento = document.getElementById('meuElemento');
8      const elementoFilho1 = document.createElement('p');
9      const elementoFilho2 = document.createElement('p');
10
11      meuElemento.appendChild(elementoFilho1);
12      meuElemento.appendChild(elementoFilho2);
13
14  </script>
```

# DOM

O append funciona da mesma forma que o appendChild, porém ele nos permite adicionar mais de um elemento filho por vez, os separando por vírgula. Utilizamos ele da seguinte forma:

```
1  <h1 id="meuElemento" class="minhaClasse">Título</h1>
2    <p class="minhaClasse">Parágrafo 1</p>
3
4    <script>
5        const meuElemento = document.getElementById('meuElemento');
6        const elementoFilho1 = document.createElement('p');
7        const elementoFilho2 = document.createElement('p');
8
9        // Adicionando os elementos filhos usando append()
10       meuElemento.append(elementoFilho1, elementoFilho2);
11    </script>
```



# DOM

O JavaScript também nos permite alterar atributos do nosso elemento, imagine o caso de um elemento `img`, onde você quer alterar o atributo `src` dele, colocando o link de uma imagem.

Nesse caso, utilizaremos o método `setAttribute`, ele vai nos permitir alterar um dos atributos daquele elemento. Primeiro vamos passar o atributo que queremos alterar, como string e depois o novo valor desse atributo. Utilizaremos ele da seguinte forma:



# DOM

Nesse caso, o atributo src do elemento com id imagem vai ser alterado para uma imagem de gatinho.



```
1 <body>
2
3 <img id="imagem" />
4 <script>
5 let ElementoImg = document.getElementById('imagem')
6 ElementoImg.setAttribute('src', 'https://www.petz.com.br/blog/wp-content/uploads/2019/04/bigode-do-gato-petz.jpg')
7
8 </script>
9 </body>
```

Também conseguimos pegar e alterar atributos do nosso elemento diretamente, basta passar .nomedoatributo e você já consegue alterar o valor. Como no exemplo a seguir:



```
1 <body>
2
3 <img id="meuElemento"></img>
4 <script>
5 let meuElemento = document.getElementById('meuElemento')
6 meuElemento.src = "https://i0.statig.com.br/bancodeimagens/2n/y9/z1/2ny9zlk3myviabfr6wd6k8ccz.jpg";
7 </script>
```



# DOM

A DOM nos traz alguns métodos para manipular a lista de classes do nosso elemento. Podemos manipular as classes do elemento através da propriedade `classList` e passando os métodos específicos para adicionar e remover classes. Podemos adicionar uma classe através do método `add`.





# DOM


No exemplo a seguir, estamos adicionando a classe **novaClasse** ao elemento.



```
1 <body>
2
3 <div id="meuElemento" class=""></div>
4 <script>
5   let meuElemento = document.getElementById('meuElemento')
6   meuElemento.classList.add('novaClasse')
7
8 </script>
```

# DOM

Podemos adicionar uma classe através do método **remove**. No exemplo a seguir, estamos removendo a classe **novaClasse** do elemento.




```
1 <body>
2
3 <div id="meuElemento" class="novaClasse"></div>
4 <script>
5 let meuElemento = document.getElementById('meuElemento')
6 meuElemento.classList.remove('novaClasse')
7
8 </script>
```

# DOM

O **innerText** vai funcionar de uma maneira similar ao **innerHTML**, nos permitindo manipular todo o texto visível de dentro daquele elemento.

Utilizamos ele da seguinte forma:




```
1 <body>
2
3 <div id="meuElemento" class="novaClasse"></div>
4 <script>
5 let meuElemento = document.getElementById('meuElemento')
6 meuElemento.innerText = 'Texto Qualquer'
7 </script>
8
```

# DOM

O **textContent** se assemelha muito ao **innerText**, porém ele vai nos permitir manipular todo o texto daquele elemento, sendo visível ou não.


Utilizaremos ele da seguinte forma:



```
1 <body>
2
3 <div id="meuElemento" class="novaClasse"></div>
4 <script>
5 let meuElemento = document.getElementById('meuElemento')
6 meuElemento.textContent = 'Texto Qualquer'
7 </script>
8
```

# DOM

A propriedade `style` vai nos permitir manipular todas as propriedades de estilo do elemento. Utilizaremos ele da seguinte forma:



```
1 <h1 id="meuElemento">Título</h1>
2 <script>
3     const meuElemento = document.getElementById('meuElemento')
4     meuElemento.style.backgroundColor = 'red'
5 </script>
```

Através dele podemos manipular qualquer propriedade de estilo, no caso anterior atualizamos a propriedade `backgroundColor` para vermelho, mas podemos alterar qualquer propriedade, como por exemplo o `fontSize`.

# DOM

Também conseguimos manipular os valores de um input, fazemos isso através da propriedade value. Segue o exemplo:

```
const entrada = document.getElementById('meuInput')
```

```
const valor = entrada.value
```

Nesse exemplo, pegamos o valor digitado pelo usuário no input e atribuímos ele a variável valor. Dessa forma, conseguimos ter acesso a dados que o usuário digitar nos formulários.



# DOM

Como mencionado anteriormente, o `querySelector` também nos permite procurar elementos que são filhos de um outro elemento. Podemos fazer isso da seguinte forma:

Nesse exemplo estamos pegando o elemento pai do elemento que queremos, e depois procurando dentro dele um elemento `p`, dessa forma temos muito mais controle de qual elemento queremos pegar e também otimizamos a página, pois dessa forma ele não irá precisar procurar o elemento na página inteira, mas apenas dentro do `elementoPai`.

```
1 <body>
2 <div id="meuElemento">
3   <p></p>
4 </div>
5 <script>
6   let meuElemento = document.getElementById('meuElemento')
7   let elementoFilho = meuElemento.querySelector('p');
8 </script>
9 </body>
```

# ATIVIDADE PRÁTICA

## **Atividade 03**

Crie um elemento através do JavaScript, adicione um texto a ele e insira ele no documento.

## **Atividade 04**

Altere o atributo src de uma tag img para trocar a imagem que ele está apresentando através do JavaScript.

# ATIVIDADE PRÁTICA

## **Atividade 05**

Pegue todos os elementos li de dentro de uma ul que tiverem a classe item de dentro de uma ul.

## **Atividade 06**

Crie uma função no JavaScript que irá trocar o texto de um elemento que está em uma lista ul.

# DESAFIO PRÁTICO

Crie uma página HTML com uma caixa de entrada (input), um botão "Adicionar Tarefa" e uma lista de tarefas (ul).

Escreva funções em JavaScript para realizar as seguintes operações:

- Adicionar uma nova tarefa à lista quando o botão "Adicionar Tarefa" for clicado.
- Marcar uma tarefa como concluída quando ela for clicada.
- Remover uma tarefa da lista quando um botão de exclusão associado a ela for clicado.

# DESAFIO PRÁTICO

Cada tarefa na lista deve ter um checkbox para indicar se está concluída e um botão de exclusão para remover a tarefa.

Adicione estilos CSS para melhorar a aparência da lista de tarefas.

Dicas:

- Use o método `document.createElement` para criar elementos (`li`, `input`, `button`) dinamicamente.

# DESAFIO PRÁTICO

- Ao marcar uma tarefa como concluída, modifique seu estilo visualmente (por exemplo, alterando a cor ou adicionando uma classe CSS).
- Considere armazenar as tarefas em uma estrutura de dados, como um array, para facilitar a manipulação.



# SE LIGA NO CONTEÚDO DA PRÓXIMA AULA!

AULA 09 DE JAVASCRIPT.  
EVENTOS DOM II

The logo consists of the letters 'IN' in a white, bold, sans-serif font, centered within a solid red square.

**INFINITY SCHOOL**  
VISUAL ART CREATIVE CENTER



# EVENTOS DOM

Os eventos DOM permitem que você crie interatividade e resposta do usuário em suas páginas da web. Quando um evento ocorre, o navegador dispara o evento correspondente e chama uma função de tratamento (callback) associada a esse evento.

Essa função de tratamento é executada para lidar com a ação ou alteração específica que ocorreu.



# EVENTOS DOM

Exemplos da **DOM** em JavaScript:

```
<body>
  <button id="meuBotao">Clique-me!</button>
  <p id="mensagem"></p>
  <script>
    // Obtém referências aos elementos de botão e parágrafo no HTML
    const botao = document.getElementById('meuBotao');
    const mensagemParagrafo = document.getElementById('mensagem');

    // Registra um evento de clique no botão
    botao.addEventListener('click', () => {
      mensagemParagrafo.innerHTML = 'Olá Raama';
    });
  </script>
</body>
```



# INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

AULA 08 - EVENTOS DOM