# Tema 13 – JavaScript HTML DOM – Parte 03

## Introdução

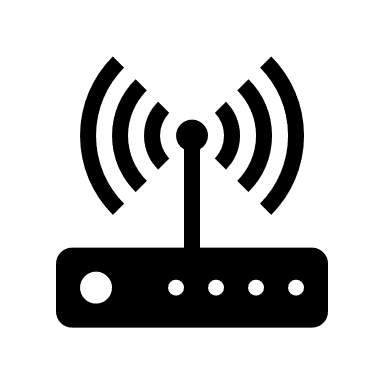
Olá, futuras mentes desenvolvedoras, sejam bem-vindos do curso de Frontend do Instituto da Oportunidade Social. Nessa aula, você aprenderá a trabalhar o Document Object Model (Modelo de objeto de documentos).

## Criação do projeto inicial

Siga os passos para criar o projeto:

1. Abra o VS Code e escolha um diretório de trabalho para o seu projeto.
2. Crie um diretório para seu projeto com o nome representativo, por exemplo, 13\_JS\_DOM\_Parte\_03.
3. Crie um arquivo dentro do diretório do projeto com o nome index.html.
4. Insira o seguinte código no seu arquivo index.html.

|  |
| --- |
| <!DOCTYPE html>  <html lang="pt-br">      <head>          <meta charset="UTF-8" />          <link rel="shortcut icon" href="#" />          <link              href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css"              rel="stylesheet"              integrity="sha384-+0n0xVW2eSR5OomGNYDnhzAbDsOXxcvSN1TPprVMTNDbiYZCxYbOOl7+AMvyTG2x"              crossorigin="anonymous"          />          <title>JavaScript com HTML DOM - Parte 03</title>      </head>      <body>          <header>              <h1 class="container text-center p-3 bg-light">                  Mais métodos do JS              </h1>          </header>          <main class="container border">              <p>Meu parágrafo sem classe.</p>              <hr />              <p class="exemplo">Meu parágrafo com a classe .exemplo</p>              <hr />              <div class="myclass">Meu elemento div com a classe .myclass</div>              <hr />              <span id="my-span">Meu elemento span com a id #my-span.</span>              <hr />              <br />              <button class="btn btn-primary" onclick="Clique()">                  Clique aqui              </button>              <hr />              <ul class="items">                  <li id="my-li" class="item">Item 1</li>                  <li class="item teste">Item 2</li>                  <li class="item teste">Item 3</li>                  <li id="my-li" class="item">Item 4</li>              </ul>          </main>          <script src="./js/main.js"></script>      </body>  </html> |

1. Esse código mostra também a marcação <script> sem nenhum código JavaScript entre a abertura e o fechamento da tag, apenas o atributo **src** com o valor main.js. Isso significa que o código JavaScript está em um arquivo externo. Portanto, temos que criar esse novo arquivo **main.js** dentro do diretório do projeto.
2. Vamos deixar o arquivo main.js vazio e à medida que vamos aprendendo coisas novas iremos completar a implementação do código JavaScript.
3. Abra o arquivo index.html, clique no botão **Go Live** da extensão **Live Server**.

## Método querySelector

O método **querySelector** retorna o p rimeiro elemento dentro do documento que corresponde ao seletor desejado. Caso nenhum elemento seja encontrado, o método retornará **null**. A sintaxe do método é:

elemento = document.querySelector(seletores);

Onde:

* elemento: é o objeto do JavaScript para armazenar o elemento encontrado.
* seletores: é uma string com um ou mais seletores desejados. Para mais de um seletor, deve-se separá-los por vírgula.

Por exemplo:

|  |
| --- |
| let paragrafo = document.querySelector("p");  let parClasse = document.querySelector("p.exemplo");  let elemento = document.querySelector(".myclass");  const myForm = document.querySelector('#my-span); |

\\\\\\\AAAAAAAAAAAAAAAAAAAA s

A primeira instrução busca o primeiro elemento parágrafo no documento. A tag entre aspas em ponto final ou # indica que queremos buscar um elemento/tag/marcação HTML.

A segunda instrução busca o primeiro elemento parágrafo que possui a classe .exemplo no documento.

A terceira instrução busca o primeiro elemento que possui a classe chamada **myclass**. O ponto final no seletor (**.myclass**) indica que queremos buscar o elemento pela classe.

A quarta instrução busca o primeiro elemento que possui o atributo **id** chamado **my-span**. A hashtag (ou tralha, ou jogo da velha, deferentes nomes para esse símbolo #) indica que desejamos procurar o elemento pela **id**.

### Vamos praticar

1. No arquivo **main.js**, insira o seguinte código.

|  |  |
| --- | --- |
| 1 | // Método querySelector  let paragrafo = document.querySelector('p');  paragrafo.style.background = '#FBBEBA'; |
| 2 |
| 3 |

1. Veja o resultado apresentado no navegador.

Graphical user interface, text, application, email

Description automatically generated

Observe que o método busca pelo elemento com a parágrafo (linha 2) do documenta. Então, esse elemento é retornado e, assim, é possível alterarmos a cor de fundo do parágrafo (linha 3). Note que o outro elemento parágrafo não teve sua cor de fundo alterado.

1. Continuando a implementação do projeto, no arquivo **main.js**, insira o seguinte código.

|  |  |
| --- | --- |
| 5 | // Busca a tag e a classe  let parClasse = (document.querySelector('p.exemplo').innerHTML =      'Mudei o texto aqui'); |
| 6 |
| 7 |

1. Veja o resultado apresentado no navegador.

Graphical user interface, text, application, email

Description automatically generated

Observe que, o texto do segundo parágrafo que possui a classe **.exemplo**. O texto do primeiro parágrafo ficou intacto. O interessante dessa instrução nas linhas 6 e 7 é que podemos fazer dois comandos de uma vez só:

* A busca do elemento com o método: **querySelector('p.exemplo')**
* A modificação do conteúdo com a propriedade: **innerHTML**.

O que foi necessário apenas é colocar os comandos na ordem certa (primeiro o método de busca e depois a propriedade) com os devidos pontos finais no ligar correto.

1. Continuando a implementação do projeto, no arquivo **main.js**, insira o seguinte código.

|  |  |
| --- | --- |
| 9 | // Busca somente a classe  let elemento = (document.querySelector('.myclass').style.fontSize = 'x-large'); |
| 10 |

1. Veja o resultado apresentado no navegador.

Graphical user interface, application

Description automatically generated

Note que, o tamanho do texto no elemento <div> ficou maior, pois ele é o elemento que contém a classe .**myclass**. Desse modo, a instrução na linha 10 busca e modifica o tamanho do texto do elemento.

## Eventos

No JavaScript, eventos são ações ou ocorrências que acontecem na página web que estamos implementando. Esses eventos geralmente invocam funções que iram executar as operações desejadas. Alguns exemplos de eventos são: o usuário clica em um botão na página, você pode querer responder a esta ação mostrando na tela uma caixa de informações ou alterando o estilo de um elemento da página ou submetendo as informações de um formulário.

### Vamos praticar

1. O nosso elemento <button> no código HTML, possui o evento **onclick** e esse evento chama a função **Clique()**, que será implementada com o JavaScript.
2. Continuando a implementação do projeto, no arquivo **main.js**, insira o seguinte código.

|  |  |
| --- | --- |
| 12 | // Função chamada pelo evento onclick  const Clique = () => {      document.querySelector('#my-span').style.backgroundColor = '#CEFBBA';  }; |
| 13 |
| 14 |
| 15 |

1. Você pode ver o resultado do evento no navegador ao clicar no botão.

Graphical user interface, text, application, email

Description automatically generated

Note que, ao clicarmos no botão, a função **Clique()** é chamada e então acontece a busca pelo elemento com a ID igual a #my-span (**querySelector('#my-span')**). Quando o elemento é encontrado a cor de fundo é alterada (**style.backgroundColor = '#CEFBBA'**).

## Método querySelectorAll

O método **querySelectorAll** retorna todos os elementos presentes no documento que possuam o seletor desejado, ou seja, esse método retorna uma lista dos elementos presentes no documento que coincidam com o seletor ou grupo de seletores especificados. A sintaxe do método é:

listaElementos = document.querySelectorAll(seletores);

Onde:

* listaElementos: é o objeto do JavaScript para armazenar os elementos encontrados.
* seletores: é uma string com um ou mais seletores desejados. Para mais de um seletor, deve-se separá-los por vírgula.

Nessa parte, vamos trabalhar com a lista que esta abaixo do botão na página web.

Graphical user interface, text, application, email

Description automatically generated

### Vamos praticar

1. Continuando a implementação do projeto, no arquivo **main.js**, insira o seguinte código.

|  |  |
| --- | --- |
| 17 | // Método querySelectorAll  let listaLi = document.querySelectorAll('li');  for (i = 0; i < listaLi.length; i++) {      listaLi[i].style.border = '1px solid #0000FF';      listaLi[i].style.padding = '3px';  }  //console.log(listaLi); |
| 18 |
| 19 |
| 20 |
| 21 |
| 22 |
| 23 |

1. Veja o resultado apresentado no navegador.

Graphical user interface, text, application

Description automatically generated

No nosso exemplo, o método **querySelectorAll** seleciona todos os elementos <li> do documento (linha 18). Se você colocar o comando **console.log(listaLi)**, poderá ver a NodeList com os quatro elementos <li> armazenados no objeto listaLi impressos no console:

Text

Description automatically generated

O laço de repetição é necessário para que possamos percorrer o array e modificar a borda (linha 20) e o espaçamento (linha 21).

1. Continuando a implementação do projeto, no arquivo **main.js**, insira o seguinte código.

|  |  |
| --- | --- |
| 25 | // Busca por id  let myLi = document.querySelectorAll('#my-li');  myLi.forEach((li) => (li.innerHTML = 'Mudou aqui!')); |
| 26 |
| 27 |

1. Veja o resultado apresentado no navegador.

Graphical user interface, text, application, email

Description automatically generated

Observe que, o método **querySelectorAll** buscar os elementos com a id igual a #my-li (linha 26) e encontra o primeiro e o último elemento da lista. Então, alteramos o conteúdo do texto desses elementos (linha 27).

Note que, ao invés de usarmos um laço de repetição, utilizamos o método de alto nível **forEach()** para percorrer o array e alterar o conteúdo através da propriedade **innerHTML**.

1. Continuando a implementação do projeto, no arquivo **main.js**, insira o seguinte código.

|  |  |
| --- | --- |
| 29 | // Busca por classe  let myItem = document.querySelectorAll('.item');  myItem.forEach((item) => (item.style.backgroundColor = '#E7BAFB')); |
| 30 |
| 31 |

1. Veja o resultado apresentado no navegador.

Graphical user interface, application

Description automatically generated

Agora, o nosso exemplo mostra o método **querySelectorAll** buscando os elementos com a classe igual a .item (linha 30) e encontra todos os elementos da lista, pois todos tem essa classe. Então, podemos alterar a cor de fundo desses elementos na linha 31.

1. Continuando a implementação do projeto, no arquivo **main.js**, insira o seguinte código.

|  |  |
| --- | --- |
| 33 | // Busca por elemento e classe  let myTeste = document.querySelectorAll('li.teste');  myTeste.forEach((elemento) => (elemento.style.color = '#FFF')); |
| 34 |
| 35 |

1. Veja o resultado apresentado no navegador.

Graphical user interface, application

Description automatically generated

Agora, o nosso exemplo mostra o método **querySelectorAll** buscando os elementos <li>, que possuem com a classe igual a .teste (linha 33) e encontra o segundo e o terceiro elementos da lista. Então, podemos alterar a cor de fonte desses elementos na linha 35.

## Para aprender mais

Procure sempre aprender e estudar mais. Seguem alguns links para você estudar e aprender mais:

querySelector():

<https://developer.mozilla.org/pt-BR/docs/Web/API/Element/querySelector>

<https://www.w3schools.com/Jsref/met_document_queryselector.asp>

Eventos:

<https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Building_blocks/Events>

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

querySelectorAll():

<https://developer.mozilla.org/pt-BR/docs/Web/API/Document/querySelectorAll>

<https://www.w3schools.com/jsref/met_document_queryselectorall.asp>