# Tema 11 – JavaScript HTML DOM – Parte 01

## Introdução

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

## DOM

O DOM (**Document Object Model**) é uma interface de programação multiplataforma e independente de linguagem, que é usada em documentos HTML, XHTML, XML e SVG. Essa interface foi criada e é mantida pela entidade World Wide Web Consortium (W3C). Basicamente, o DOM fornece uma representação estruturada do documento como uma árvore, onde os elementos do documento são como nós das ramificações da árvore. Para um documento HTML, a Figura 1 mostra a árvore DOM com alguns elementos (tag). É importante você ver essa árvore na língua inglesa mesmo, pois o HTML e o DOM são baseados no inglês e você vai usar essas palavras na língua original desses recursos.

Text

Description automatically generated

Figura 1. Árvore DOM no HTML

O DOM define métodos/funções que permitem acesso a essa árvore, de modo que permita alterar a estrutura, estilo e conteúdo do mostrado no documento. Não somente o documento pode ser manipulado como também os nós (elementos) podem ser modificados através de eventos. Desse modo, a essência do DOM é conectar páginas web a scripts ou a linguagens de programação.

Toda página carregada no navegador possui o objeto **document**, que está topo da árvore e é o único objeto nesse nível (Figura 1). Esse objeto serve como ponto de entrada para o conteúdo da página, ou seja, ele permite utilizarmos um script ou uma determinada linguagem de programação para manipular os elementos existentes nessa página.

## JavaScript HTML DOM

Até o momento, utilizamos o console para exibir as mensagens com o resultado do processamento de alguma informação no JavaScript. O console é importante para você fazer o *debug* de um código, aprender lógica de programação de uma determinada linguagem e ser um recurso de visualização do estado de uma aplicação, mas o Javascript é muito mais do isso. A principal aplicação do JavaScript é realizar a manipulação dos elementos do DOM, pois o JavaScript foi criado para incorporar uma lógica computacional dentro de um ambiente de script, que é o ambiente encontrado em um navegador web. Pois o navegador web consegue executar um script, mas não consegue executar um programa.

Com o DOM, Javascript tem o poder para criar conteúdo dinâmico em um documento HTML, permitindo o JS:

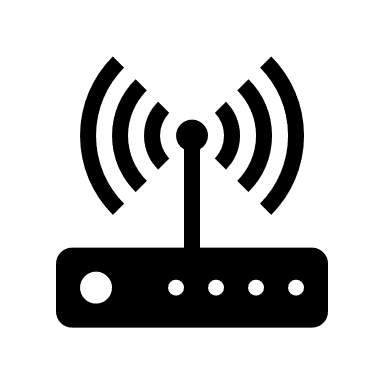
* alterar todos os elementos HTML na página,
* alterar todos os atributos HTML na página,
* mudar todos os estilos CSS na página,
* remover elementos e atributos HTML existentes,
* adicionar novos elementos e atributos HTML,
* reagir a todos os eventos HTML existentes na página e
* pode criar novos eventos HTML na página.

## Objeto window

O objeto window representa a janela do navegador que contém o elemento DOM. Foi esse objeto que permitiu utilizarmos o método alert() para exibir mensagens no navegador no Tema 01. Vamos fazer um pequeno projeto para ver esse objeto funcionando. Vamos criar o script no arquivo HTML mesmo, pois é somente uma demonstração didática. 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, 11\_JS\_DOM\_Parte\_01.
3. Crie um arquivo dentro do diretório do projeto com o nome teste.html.
4. Insira o seguinte código no seu arquivo teste.html.

|  |  |
| --- | --- |
| 1 | <!DOCTYPE html>  <html lang="pt-br">      <head>          <meta charset="UTF-8" />          <link rel="shortcut icon" href="#" />          <title>Objeto Window</title>      </head>      <body>          <script>              // objeto window              console.log(window);              window.alert('Olá alunos');              alert('Olá alunos2');          </script>      </body>  </html> |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |
| 10 |
| 11 |
| 12 |
| 13 |
| 14 |
| 15 |
| 16 |
| 17 |

1. Clique no botão **Go Live** da extensão **Live Server** e abra as Ferramentas de desenvolvimento do navegador web (Atalho **F12**). O resultado é mostrado nas imagens abaixo.

Graphical user interface, text, application

Description automatically generated

Primeiro o navegador mostra a primeira mensagem, que foi gerada pela instrução na linha 13.

Graphical user interface, text, application

Description automatically generated

Após clicar em OK na primeira mensagem, o navegador mostra a segunda mensagem, que foi gerada pela instrução na linha 14.

Isso mostra que o objeto window não precisa estar explicito na chamada dos seus métodos.

|  |
| --- |
| window.alert('Olá alunos');  alert('Olá alunos2'); |

Se abrirmos o console para ver o que foi impresso com a instrução na linha 11, podemos ver o objeto **window** impresso no console.

Graphical user interface, text, application

Description automatically generated

Se expandirmos esse objeto, podemos ver mais detalhes dele com suas propriedades de métodos para manipularmos a janela de exibição do documento. Não se preocupe se as informações desse objeto parecerem diferentes para você, pois essas informações mudam de navegador para navegador.

Text

Description automatically generated with medium confidence

Alguns dos métodos do objeto **window** são mostrados abaixo:

|  |  |
| --- | --- |
| **Method** | **Description** |
| **alert()** | Mostra uma caixa de alerta contendo uma mensagem e um botão de OK. |
| **confirm()** | Mostra uma caixa de diálogo contendo uma mensagem com um botão de OK e um de CANCEL. |
| **prompt()** | Mostra uma caixa de diálogo para o usuário entrar com algum dado. |
| **open()** | Abre uma nova janela no navegador. |
| **close()** | Fecha a janela atual do navegador. |
| **setTimeout()** | Executa uma ação após um determinado tempo. Essa ação pode ser uma chamada de função, uma expressão de avaliação, etc. |

## Métodos do DOM

Como foi dito toda página possui um objeto **document** e ele é a porta de acesso para todos os elementos da página web. Desse modo, é possível pelo JavaScript utilizar os diversos métodos do DOM.

### Vamos praticar

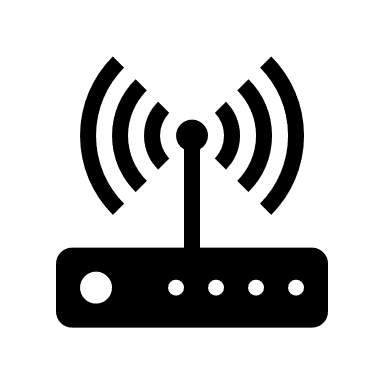
Vamos ver no console, algumas das informações que esse objeto carrega. Siga os passos para criar o projeto:

1. Abra o VS Code e escolha um diretório de trabalho para o seu projeto.
2. No diretório 11\_JS\_DOM\_Parte\_01, crie um arquivo dentro do diretório do projeto com o nome index.html e 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="#" />          <title> JavaScript com HTML DOM – Parte 01</title>      </head>      <body>          <script src="./js/main.js"></script>      </body>  </html> |

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

|  |  |
| --- | --- |
| 1 | // JavaScript e DOM  // objeto document  console.log(document); |
| 2 |
| 3 |

1. Abra o arquivo index.html, clique no botão **Go Live** da extensão **Live Server** e abra as Ferramentas de desenvolvimento do navegador web (Atalho **F12**). Veja o objeto impresso no console.

Graphical user interface, text, application

Description automatically generated

Se expandirmos o objeto HTMLDocument, podemos ver algumas das informações carregados por ele.

Text

Description automatically generated

Só esse pedaço mostrado no podemos ver diversas propriedades e elementos como: URL, head, body, doctype, etc. Vamos testar acessar algumas dessas informações.

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

|  |  |
| --- | --- |
| 5 | // Acessar informações  console.clear();  console.log(document.domain);  console.log(document.URL);  console.log(document.title);  console.log(document.head);  console.log(document.body); |
| 6 |
| 7 |
| 8 |
| 9 |
| 10 |
| 11 |

1. Veja as informações impressas no console.

Graphical user interface, text, application, email

Description automatically generated

A instrução na linha 7 imprime o domínio que o documento está aberto:



A instrução na linha 8 imprime URL que o documento está aberto:



A instrução na linha 9 imprime o título do documento:



A instrução na linha 10 imprime o elemento <head> do documento:



A instrução na linha 11 imprime o elemento <body> do documento:



Podemos alterar qualquer informação se quisermos. Por exemplo, vamos alterar o título da página e criar um elemento <h1> no body da página.

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

|  |  |
| --- | --- |
| 13 | // Alterar o título  document.title = 123;  // Criar um elemento <h1>  let heading = document.createElement('H1');  heading.innerHTML = 'Olá alunos!';  document.body.appendChild(heading);  cabecalho.style.borderBottom = 'solid 3px #000'; |
| 14 |
| 15 |
| 16 |
| 17 |
| 18 |
| 19 |

1. Vamos ver o resultado mostrado no console.

Graphical user interface, text, application

Description automatically generated

Observe que a instrução na linha 14, altera o título da página para 123. E as instruções nas linhas 16 até 18 criam a tag <h1> e inserir um conteúdo dentro desse elemento para então ser exibido dentro do <body> da página.

O método **createElement**() cria um novo nó/elemento com um nome específico. Esse nome de pode ser com letras maiúsculas ou minúsculas.

|  |  |
| --- | --- |
| Criar um elemento botão | document.createElement('BUTTON'); |
| Criar um elemento parágrafo | document.createElement('p'); |
| Criar um elemento imagem | document.createElement('img'); |

A propriedade **innerHTML** define ou retorna o conteúdo de um elemento HTML.

E o método **appendChild**() adiciona o nó/elemento criado em um elemento pai específico.

document.body.appendChild(heading);

No exemplo, o elemento **heading** foi adiciona dentro do elemento **<body>**.

A propriedade **style** na linha 19 permite configura o estilo CSS desse novo elemento.

Se usarmos o recurso de Elements (no navegador Chrome) ou Inspetor (no navegador Firefox), podemos ver novo elemento inserido no código HTML da página.

Graphical user interface, text, application

Description automatically generated

## Para aprender mais

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

DOM:

<https://pt.wikipedia.org/wiki/Modelo_de_Objeto_de_Documentos>

<https://developer.mozilla.org/pt-BR/docs/Glossary/DOM>

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

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

Objeto window:

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

Objeto document:

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

Método createElement():

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

Propriedade innerHTML:

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

Método appendChild():

<https://developer.mozilla.org/pt-BR/docs/Web/API/Node/appendChild>