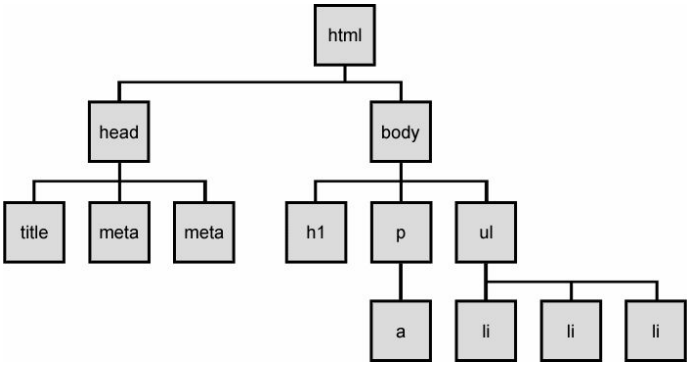
Aula 2 - Document Object Model I (Acessando e alterando elementos)

- Introdução
- O que é o DOM
- Estrutura de Árvore
- Acessando elementos
- Acessando filhos, pais e irmãos
- Alterando elementos

Nesta aula vamos aprender sobre como podemos editar um documento HTML de forma dinâmica, ou seja, como podemos alterar um documento depois dele ter sido criado, usando JavaScript!

O que é o DOM

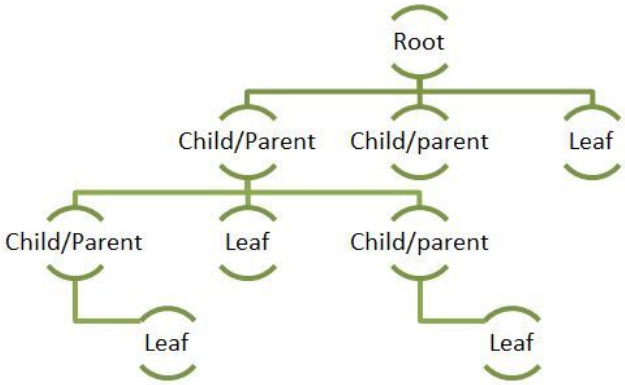
O Document Object Model não é algo exclusivo do JavaScript. Ele é uma interface oferecida pelo navegador para que possamos editar uma página dinamicamente usando louma linguagem de programação. O DOM é basicamente um modelo padronizado para a manipulação de arquivos HTML.



Estrutura de Árvore

O DOM utiliza uma estrutura de dados de Árvore para criar o modelo de um documento HTML. Vamos analisar algumas características importantes dessa estrutura para facilitar nosso entendimento do DOM:

Uma árvore é uma coleção de **nós** que estão distribuídos de forma hierárquica. Por padrão, um nó de uma árvore tem um valor, **filhos** (nós que estão logo abaixo dele) e um **pai** (nó logo acima dele). Porém, existe também a **raíz** (root), que é o único nó que não tem pai, e as **folhas** (leaves), que são os nós que não têm filhos (são os que ficam nas extremidades da árvore).



É importante conhecer essas características da estrutura de árvore porque para realizar o trabalho de manipular esses elementos HTML teremos que navegar pela árvore, acessando seus nós para fazer as alterações que queremos. Vamos ver agora como fazemos isso.

Acessando Elementos

Utilizando o DOM, podemos acessar qualquer nó da árvore utilizando alguns métodos em elementos que a sua interface traz para nós. Esses são:

1 - document.getElementById(id)
2 - [elemento].getElementsByClassName()
3 - [elemento].getElementsByTagName()
4 - document.getElementsByName()
5 - [elemento].querySelector()
6 - [elemento].querySelectorAll()

document.getElementById(id)
Através desse método conseguimos selecionar um elemento HTML pelo seu atributo id.

```
    <button id="botao">

    <script>
        let elementoBotao = document.getElementById('botao');
        console.log(elementoBotao) // <button id="botao">
    </script>
```

OBS.: Esse método só pode ser usado no elemento document, que é o elemento raiz.

document.getElementsByName(name)
Através deste método conseguimos selecionar uma coleção de elementos HTML pelo seu atributo name.

```
<form>
    <input name="nome" type="text">
    <input name="idade" type="text">
    <input name="genero" value="masculino" type="radio">
    <input name="genero" value="feminino" type="radio">
</form>

<script>
    let inputsGenero = document.getElementsByName('genero');
    console.log(inputsGenero) // NodeList[input,input]
</script>
```

OBS1.: Este método retorna uma NodeList com todos os elementos HTML selecionados. Uma NodeList é um objeto semelhante a uma array que serve para guarda nós (seja nós de elemento, texto ou comentário)

OBS2.: Esse método só pode ser usado no elemento document, que é o elemento raiz.

[elemento].getElementsByClassName(class)
Através deste método conseguimos selecionar uma coleção de elementos HTML pelo seu atributo class dentre os filhos do elemento que executa o método.

```
<div class="botoes">
    <button class="sucesso"></button>
    <button class="erro"></button>
    <button class="arredondado sucesso"></button>
    <button class="arredondado erro"></button>
</div>

    <script>
        let botoesArredondados = document.getElementsByClassName('arredondado');
        console.log(botoesArredondados) // HTMLCollection[button.arredondado.sucesso, button.arredondado.erro]
    </script>
```

OBS1.: Este método retorna uma HTMLCollection com todos os elementos HTML selecionados. Diferente da NodeList, ela traz somente nós que sejam de elementos.

[elemento].getElementsByTagName(tag)
Através deste método conseguimos selecionar uma coleção de elementos HTML pelo nome da sua tag (nome do elemento) dentre os filhos do elemento que executa o método.

```
<div class="campos-de-dados">
    <label>Nome</label>
    <input type="text">
</div>

    <script>
        let elementoInput = document.getElementsByTagName('input');
        console.log(elementoInput); //HTMLCollection[input]
    </script>
```

[elemento].querySelector(seletor)
Através deste método conseguimos selecionar um elemento dentre os filhos do elemento que executa o método utilizando um seletor de CSS.

```
<ul>
    <li>
        <button></button>
    </li>
    <li>
        <input type="text">
    </li>
    <li>
        <p>Lorem ipsum dolor sit amet</p>
    </li>
</ul>

    <script>
        let elementoParagrafo = document.querySelector('ul > li:nth-child(3)');
        console.log(elementoParagrafo); //<li><p>Lorem ipsum dolor sit amet</p></li>
    </script>
```

[elemento].querySelectorAll(seletor)
Através deste método conseguimos selecionar uma coleção de elementos dentre os filhos do elemento que executa o método utilizando um seletor de CSS.

```
<ul>
    <li>
        <button></button>
    </li>
    <li>
        <input type="text">
    </li>
    <li>
        <p>Lorem ipsum dolor sit amet</p>
    </li>
</ul>

    <script>
        let elementosItensLista = document.querySelectorAll('ul > li');
        console.log(elementosItensLista); // NodeList[li,li,li]
    </script>
```

Acessando filhos, pais e irmãos

Também é possível navegar pelo DOM através das relações dos nós para conseguir chegar a algum nó específico através de alguns métodos. Vamos utilizar esta tabela em HTML como exemplo para mostrar como os métodos são utilizados:

```
<table>
    <tr id="cabecalho">
        <th>Nome</th>
        <th>Tipo</th>
        <th>Região</th>
        <th>Evolução</th>
    </tr>
    <tr id="nome-pokemon">
        <td>Vibrava</td>
        <td>Piplup</td>
        <td>Taillow</td>
    </tr>
    <tr id="tipo-pokemon">
        <td>Dragão/Terrestre</td>
        <td>Água</td>
        <td>Voador/Normal</td>
    </tr>
    <tr id="regiao-pokemon">
        <td>Hoenn</td>
        <td>Sinnoh</td>
        <td>Hoenn</td>
    </tr>
    <tr id="evolucao-pokemon">
        <td>Flygon</td>
        <td>Prinplup</td>
        <td>Swellow</td>
    </tr>
</table>
```

[elemento].children
Retorna uma HTMLCollection com todos os filho do elemento (os filho no HTMLCollection podem ser acessados assim como se acessa um array normal, com *"[índice]"*).

```
let linhaNomesPokemon = document.querySelector('tr#nome-pokemon');
let filhosNomesPokemon = linhaNomesPokemon.children;
console.log(filhosNomesPokemon); //HTMLCollection [td,td,td]
```

[elemento].firstElementChild
Acessa diretamente o primeiro filho do elemento

[elemento].lastElementChild
Acessa diretamente o último filho do elemento

[elemento].parentElement
Retorna o elemento pai desse elemento.

```
let cabecalhoPokemon = document.querySelector('tr#cabecalho');
let tabela = cabecalhoPokemon.parentElement
console.log(tabela); //<table>
```

[elemento].nextElementSibling
Retorna o próximo irmão do elemento.

```
let linhaTipoPokemon = document.getElementById('tipo-pokemon');
let linhaRegiaoPokemon = linhaTipoPokemon.nextElementSibling;
console.log(linhaRegiaoPokemon); //<tr id="regiao-pokemon">
```

[elemento].previousElementSibling
Retorna o irmão anterior do elemento.

```
let linhaTipoPokemon = document.getElementById('tipo-pokemon');
let linhaNomePokemon = linhaTipoPokemon.previousElementSibling;
console.log(linhaNomePokemon); //<tr id="nome-pokemon">
```

Alterando elementos

Tendo selecionado nossos elementos, podemos fazer uso de diversas propriedades presentes neles para alterá-los como:

[elemento].style
Para escolher um estilo CSS para mudar nele.

```
let linhaTipoPokemon = document.getElementById('evolucao-pokemon');
linhaTipoPokemon.style.backgroundColor = 'red';
linhaTipoPokemon.style.fontFamily = 'cursive';
linhaTipoPokemon.style.fontSize = '20pt';
```

[elemento].textContent
Para mudar o conteúdo de texto de dentro do elemento.

```
let linhaTipoPokemon = document.getElementById('evolucao-pokemon');
let linha1 = linhaTipoPokemon.firstElementChild;
linha1.textContent = 'Outro Pokémon';
```

E foi isso para a aula de hoje! Se ficou com dúvidas de como alguns métodos funcionam, tente criar um documento HTML com essa mesma tabela e executar cada um dos comandos pra ver o que acontece.

Na próxima aula vamos descobrir como podemos mexer mais ainda com os elementos de um documento através do DOM!