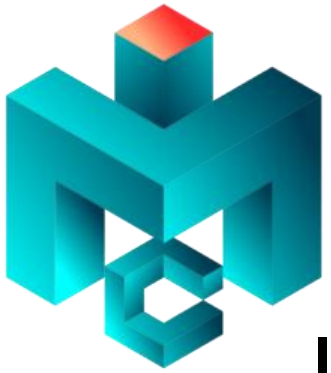


Aula – 8
Introdução ao DOM (Document Object Model)

Disciplina: XDES03 – Programação Web

Prof: Phyllipe Lima Francisco
phyllipe@unifei.edu.br

Universidade Federal de Itajubá – UNIFEI
IMC – Instituto de Matemática e Computação

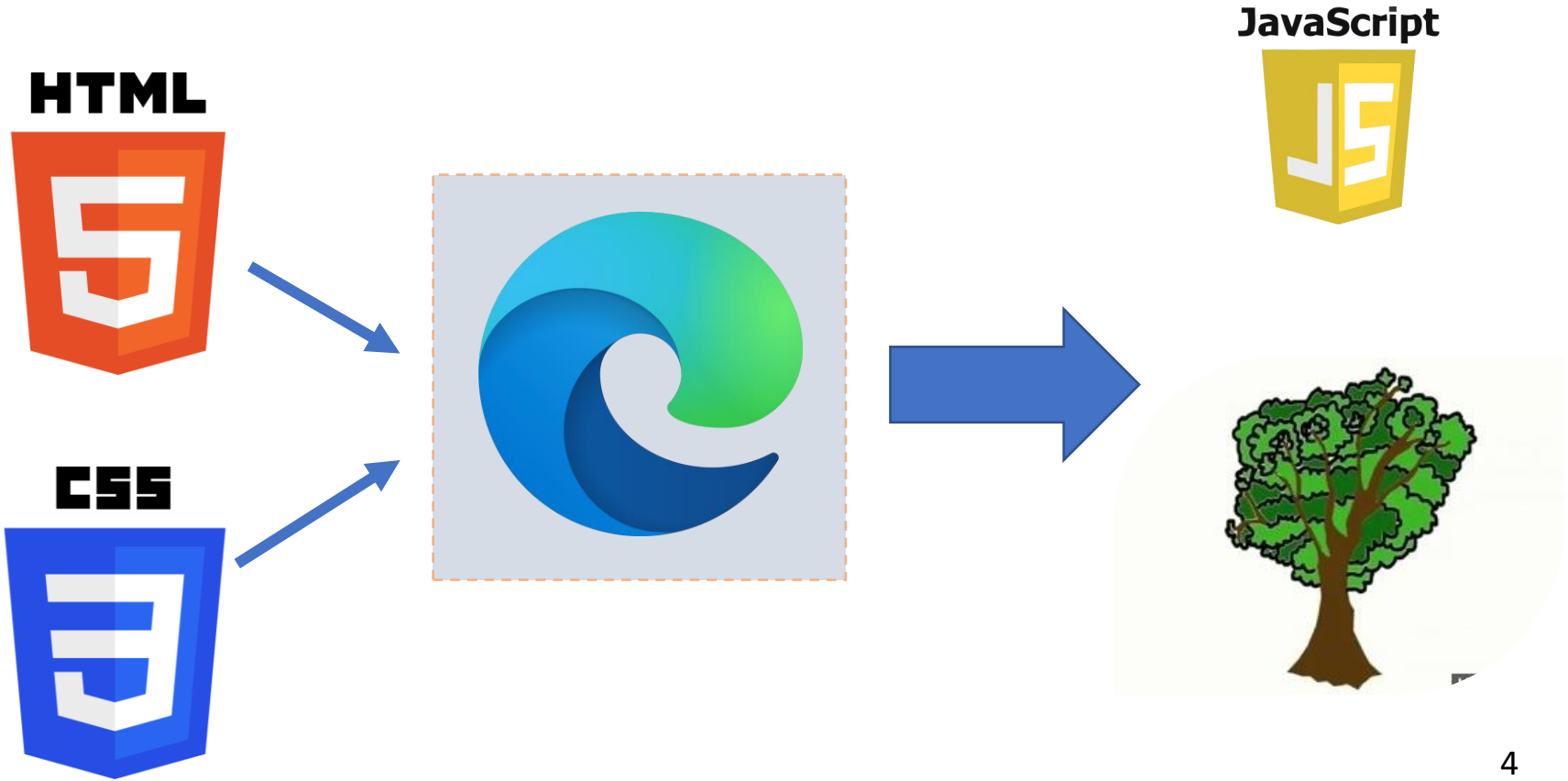
Agenda

- O que é o DOM
- Buscando elementos no DOM
- Buscando conteúdo dos elementos
- Modificando estilização
- Adicionando Elementos
- Removendo Elementos

Navegador Cria o DOM

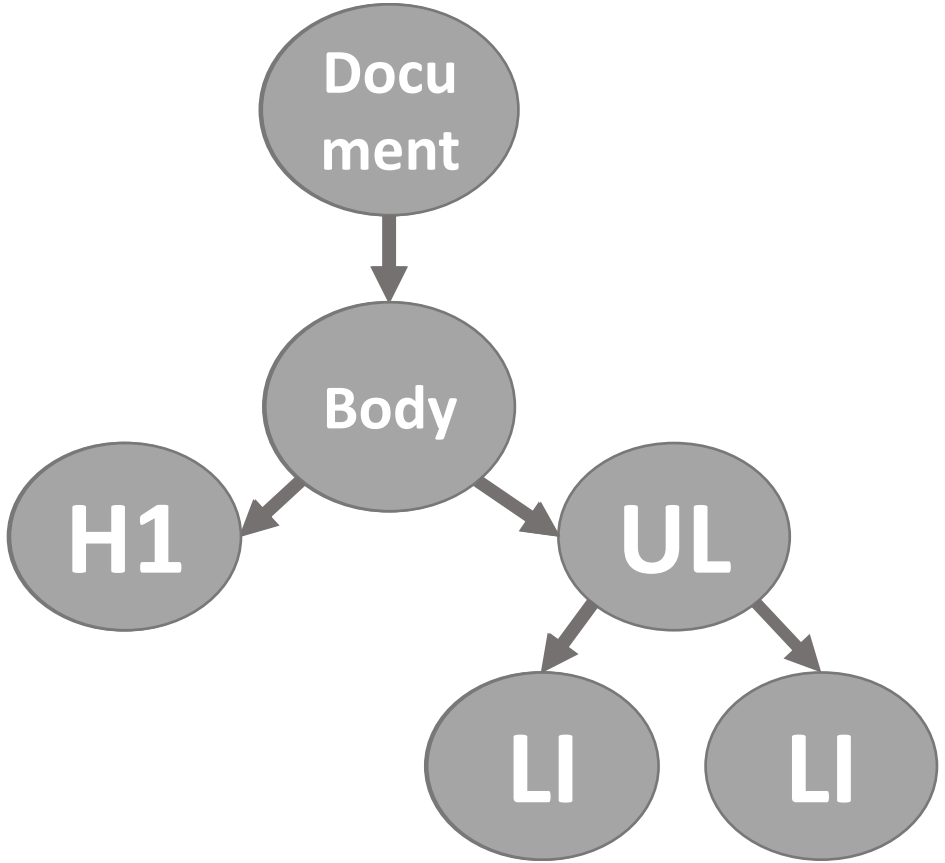
- Durante o processo de renderizar e *loading* da página WEB, o navegador transforma cada elemento em um objeto JS.
- Esses objetos são armazenados em uma estrutura de árvore.
- Essa árvore é o DOM.

Navegador Cria o DOM



DOM - Árvore

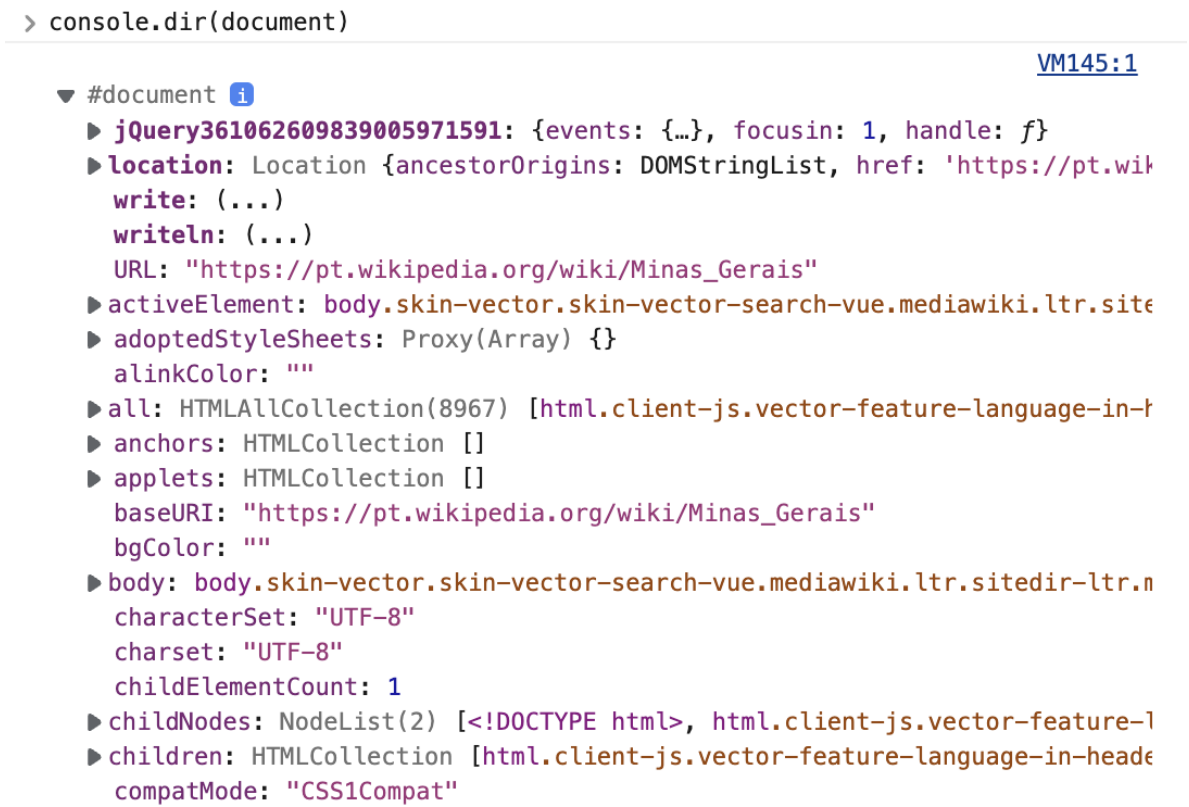
```
<body>
    <h1>Tarefas da semana</h1>
    <ul>
        <li>Estudar Web</li>
        <li>Estudar C++</li>
    </ul>
</body>
```



Visualizar o DOM

- É possível invocar o objeto DOM diretamente no JS com *document*.
- O comando console.dir(document) apresenta toda a estrutura da página em formato objeto Javascript

Exemplo DOM Wikipedia



Seleção de Elementos

Por que selecionar elementos?

Em muitos cenários desejamos mudar informações de elementos HTML específicos, por isso precisamos selecionar.

Como fazer a seleção?

Originalmente, essa seleção poderia ser feita por *id, class, tag* e *name* (o atributo)

O que é retornado?

O retorno é um objeto JavaScript com as propriedades do elemento HTML

Selecionando pelo ID

- Document.getElementById();
- Esse método devolve o elemento com a ID especificada.
- Caso nenhum elemento seja encontrado, o valor atribuído é *null*.

Selecionando pelo ID – Exemplo

```
const cabecalhoNull = document.getElementById('cabecalho');
//null
const cabecalhoCorreto = document.getElementById('heading');
//O id correto, nesse exemplo, é heading
//console.log irá imprimir o HTML
//<h1 id="heading">Calculadora Moderna Ultra Megazorde</h1>
```

Selecionando pela *Class*

- Document.getElementsByClass();
- Esse método devolve um objeto do tipo *HTMLCollection*, que pode ser visto como Array com os elementos HTML.
- Apesar de ser indexada e permitir iteração não é um Array convencional com as funções da API Array

Selecionando pela *Class* – Exemplo 1

- Selecionando todos os botões de operação da calculadora

```
const botoesOP = document.getElementsByClassName('calculadora-tecla-operador');

for(let botao of botoesOP)
        console.log(botao);//os elementos HTML serão impressos
```

Selecionando pela *Class* – Exemplo 2

- Imprimindo todos os valores do atributo “type” para cada elemento.

```
const botoesOP = document.getElementsByClassName('calculadora-tecla-operador');

for(let botao of botoesOP)
        console.log(botao.type);//o valor “button” será impresso
```

Selecionando pela *Class* – Exemplo 3

- Transformando todos os botões de operação em soma (+)

```
const botoesOP = document.getElementsByClassName('calculadora-tecla-operador');

for(let botao of botoesOP)
        botao.innerText = '+';//innerText acessa o conteúdo visivel
```

Selecionando por *Tags*

- Document.getElementsByTagName.
- Retorna uma *HTMLCollection* com todos os elementos da *tag* selecionada.
- Se nenhum elemento for encontrado, é retornada a coleção vazia, ao invés de *null*.

Selecionando por *Tags* - Exemplo

- Selecionando todos os botões.

```
const botoes = document.getElementsByTagName('button');

for(let botao of botoes)
        console.log(botao);//todos os botões serão impressos
```

Tem como selecionar semelhante ao CSS?

Sim! A forma mais utilizada e moderna para buscar elementos no DOM é com *querySelector*.

querySelector - Definição

- Com o avanço do JS, um novo método foi adicionado ao DOM.
- Este permite fazer a seleção da mesma forma que é feita no CSS.
- É possível buscar por id, class, atributos, e todas as outras combinações usadas no CSS.

querySelector - Exemplos

- document.querySelector('#id’);
- document.querySelector(‘.class’);
- document.querySelector(‘nome-tag’);
- Em todos esses casos, o retorno é apenas um elemento. Para retornar uma lista, se usa o método querySelectorAll().

querySelector – Exemplos ID

- Retornando elemento H1 por ID.

```
const h = document.querySelector("#heading");
console.log(h);
```

querySelector – Exemplos Class

- Retornando o primeiro botão com a classe especificada

```
const btn = document.querySelector(".calculadora-tecla-operador");
```

querySelector – Exemplos Tag

- Retornando o primeiro botão buscando pela *tag*.

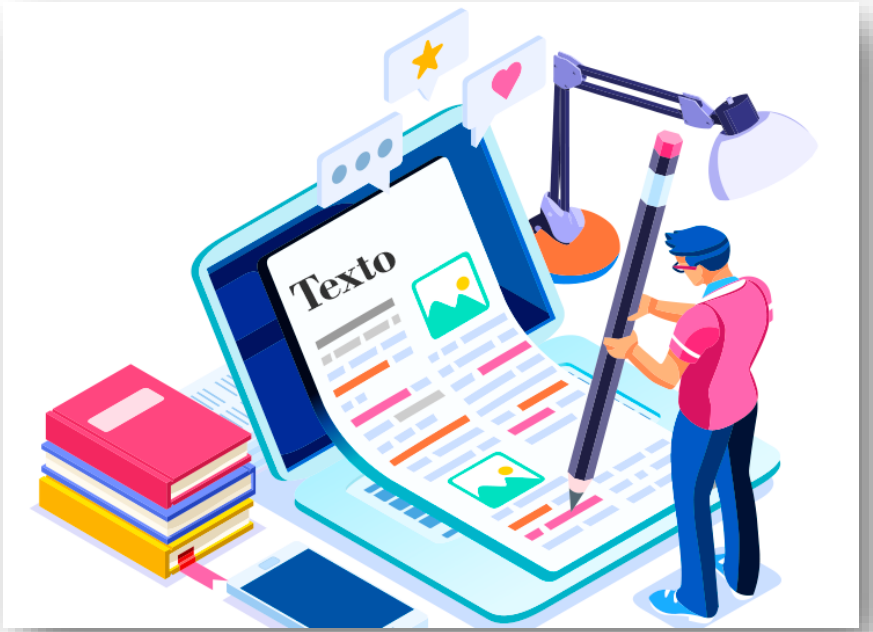
```
const elemento = document.querySelector("button");
```

querySelectorAll – Exemplos

- Retornando todos os botões com a classe especificada.

```
const btnS = document.querySelectorAll('.calculadora-tecla-operador');
console.log(btnS.length);//4
```

Conteúdo Textual



Conteúdo Textual

- Podemos manipular o conteúdo textual dentro dos elementos HTML com as seguintes propriedades:
  - innerHTML
  - innerText
  - textContent

Conteúdo Textual - innerText

- Retorna o texto visível que se encontra no elemento HTML.
- Se algum texto estiver com a condição ‘hidden” não será mostrado

Conteúdo Textual - textContent

- Retorna o texto completo que se encontra no elemento HTML, mesmo que esteja “Hidden”.

Conteúdo Textual – Exemplo innerText e textContent

- HTML Exemplo:

```
<p id="paragrafo">
  Eu sou um paragrafo.
  <span style="display: none;">Eu estou
hidden.</span>
</p>
```

- JavaScript:

```
//Eu sou um paragrafo.
document.querySelector('#paragrafo').innerText;

//Eu sou um paragrafo. Eu estou hidden.
document.querySelector('#paragrafo').textContent
```

Conteúdo Textual - innerHTML

- Retorna o texto completo que se encontra no elemento HTML, incluindo HTML interno, isto é, elementos HTML descendentes.

Conteúdo Textual – innerHTML Exemplo

```
<p id="paragrafo">
  Eu sou um paragrafo.
  <span style="display: none;">Eu estou
hidden.</span>
</p>
```

```
//Eu sou um paragrafo. <span style="display: none;">Eu estou hidden.</span>
document.querySelector('#paragrafo').innerHTML;
```

Conteúdo Textual - Modificações

- Com essas propriedades é possível também alterar o conteúdo e com isso sobrescrever utilizando o operador de atribuição.

```
document.querySelector('#paragrafo').innerText = 'Novo Texto';
document.querySelector('#paragrafo').innerText += ' Concatenando';
```

Acessando os Atributos

Acessando Atributos

- Os atributos dos elementos HTML podem ser acessando como propriedades do objeto JS. Pode-se utilizar o operador ponto (.) para acesso.
- Esse cenário é interessante quando se está criando os elementos, e deseja preencher os valores dos atributos.

Acessando Atributos - Métodos

- É possível também utilizar dois métodos para acessar e modificar os atributos.
- getAttribute() para acessar
- setAttribute() para modificar

Acessando Atributos - Exemplo

- Modificar a descrição da imagem, isto é, o atributo 'alt'. Usando as duas formas.

```
const mapaMG = document.querySelector('#minas-gerais img’);

mapaMG.alt = 'O mapa de minas gerais';

mapaMG.setAttribute('alt', 'O mapa de minas gerais');
```

Estilizando

Modificando as propriedades do CSS - inline

- As propriedades de estilização podem ser modificadas diretamente no objeto JS, mas estas estão escritas em formato *Camel Case*.
- As propriedades inseridas via JS serão *inline* e pode não ser conveniente adicionar nesse formato.
- A propriedade *style* é acessada primeiro.

Modificando as propriedades do CSS - Exemplo

- Cada propriedade é acessada separadamente

```
const pp = document.querySelector('#paragrafo');
pp.style.fontSize = '5em';
pp.style.color = 'blue';
```

Acessa as propriedades do CSS

- As estilizações que não são inline, não conseguem ser lidas diretamente pelo propriedade *style*
- É preciso acessar o objeto global *window* e o método *getComputedStlye()* passando como parâmetro o objeto JS.
- Com isso é possível acessar as propriedades.

Acessa as propriedades do CSS - Exemplo

```
const pp = document.querySelector('#paragrafo');
pp.style.color = 'red’;

window.getComputedStyle(pp).color;//rgb(255, 0, 0)

window.getComputedStyle(ppp).fontSize;// 16px
```

Exercício 1

- Crie um código HTML/JS que escreva a frase RAINBOW com diferentes cores. Coloque o texto em um H1 e cada letra com uma *span*
- Como sugestão usem as seguintes cores:

```
const cores = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet'];
```

Exercício 1 - HTML

- Use o HTML fornecido

```
<body>
        <h1>
            <span>R</span>
            <span>A</span>
            <span>I</span>
            <span>N</span>
            <span>B</span>
            <span>O</span>
            <span>W</span>
        </h1>
    </body>
```

Modificando com *classList*

- Uma forma mais interessante de estilizar a partir do JS é manipulando as classes CSS.
- Para isso usamos a propriedade *classList*
- Esta apresenta uma API que permite adicionar, remover e ler as classes utilizadas no elemento HTML.

Modificando com *classList*

- classList.add()
- classList.remove()
- classList.toggle() // desliga/liga a classe

```
const pp = document.querySelector('#paragrafo');
pp.classList.add('caixa-p');
```

Percorrendo a Árvore DOM

- parentElement
  - Propriedade que devolve o elemento pai.
- childElementCount
  - Retona o número de descendentes.
- children (HTMLCollection)
  - Retorna uma coleção com os descentes.

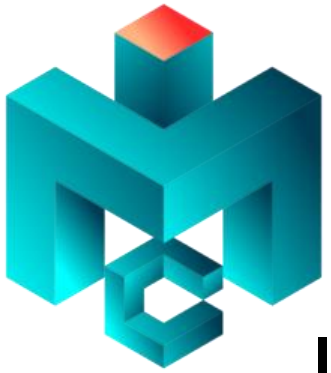
Adicionando Elementos

- createElement(“nome tag”)
- append(“elemento ou texto”)
- appendChild(“apenas elementos”)
- prepend(“elemento ou texto”) //add no início
- inserAdjacentElement()//adiciona adjacente
- after()//Adiciona após

Removendo Elementos

- Element.remove();

```
const img = document.querySelector('img');
img.remove();
```



Aula – 8
Introdução ao DOM (Document Object Model)

Disciplina: XDES03 – Programação Web

Prof: Phyllipe Lima Francisco
phyllipe@unifei.edu.br

Universidade Federal de Itajubá – UNIFEI
IMC – Instituto de Matemática e Computação