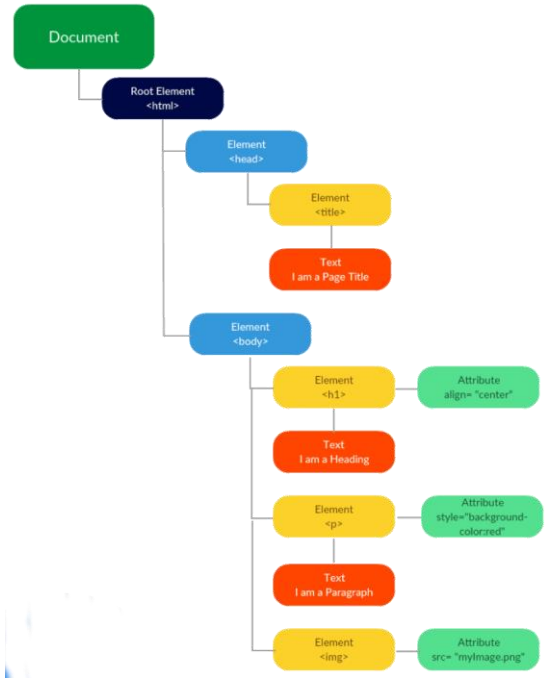
Curso Superior de Tecnologia em Sistemas para Internet
Desenvolvimento de Sites para Web (DSW)
Rafael Cunha Cardoso
rafaelcardoso@ifsul.edu.br



DOM – Document Object Model

INSTITUTO FEDERAL
Sul-rio-grandense
Câmpus Pelotas

DOM – *Document Object Model*

Conjunto de objetos dentro do navegador que provê acesso aos componentes de um site.

Quando uma página web é carregada, **o navegador cria o DOM da página**.

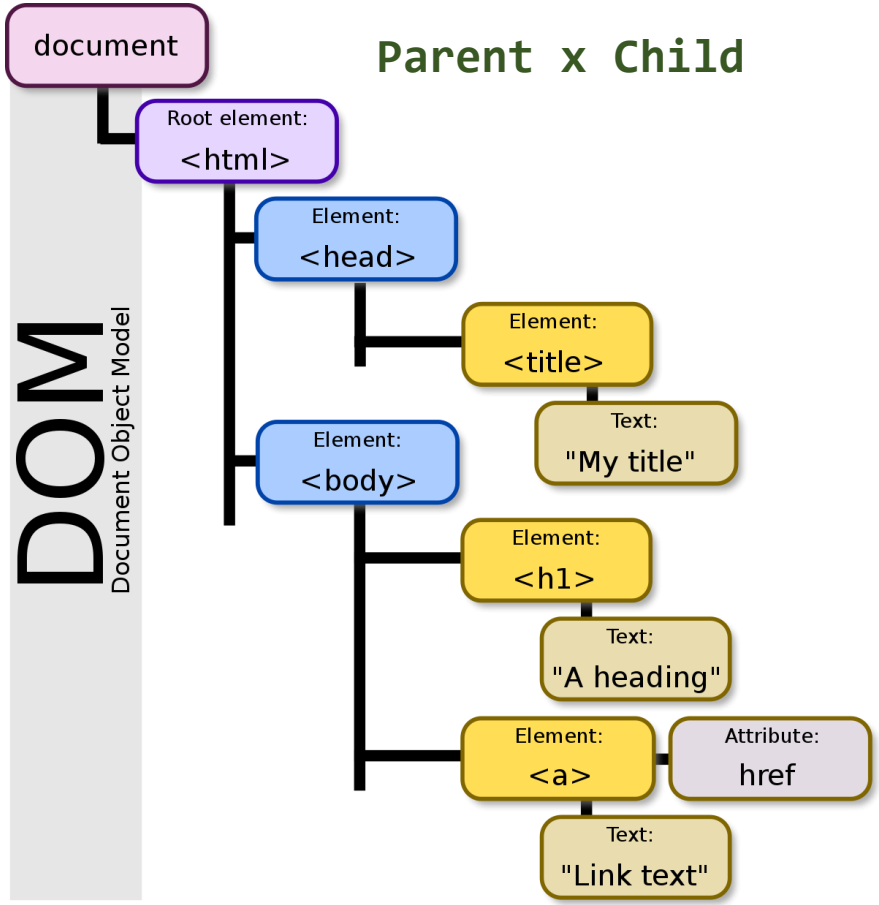
Segundo **W3C**:

*"O DOM do W3C é uma interface independente de plataforma e linguagem que permite aos programas e scripts acessar e atualizar dinamicamente a **estrutura**, o **conteúdo** e a **estilização** de documentos."*

Objetivo: Simplificar a tarefa de **acessar e manipular o documento**.

DOM API

API de programação para os documentos HTML e XML, com a finalidade de auxiliar o desenvolvimento de aplicações gerais.



Parent x Child

Representa a página Web, **por meio de nós e objetos**, de forma que os programas possam **alterar a estrutura, o estilo e o conteúdo documento**.

Quando uma página Web é carregada, o navegador cria um modelo de objeto de documento da página

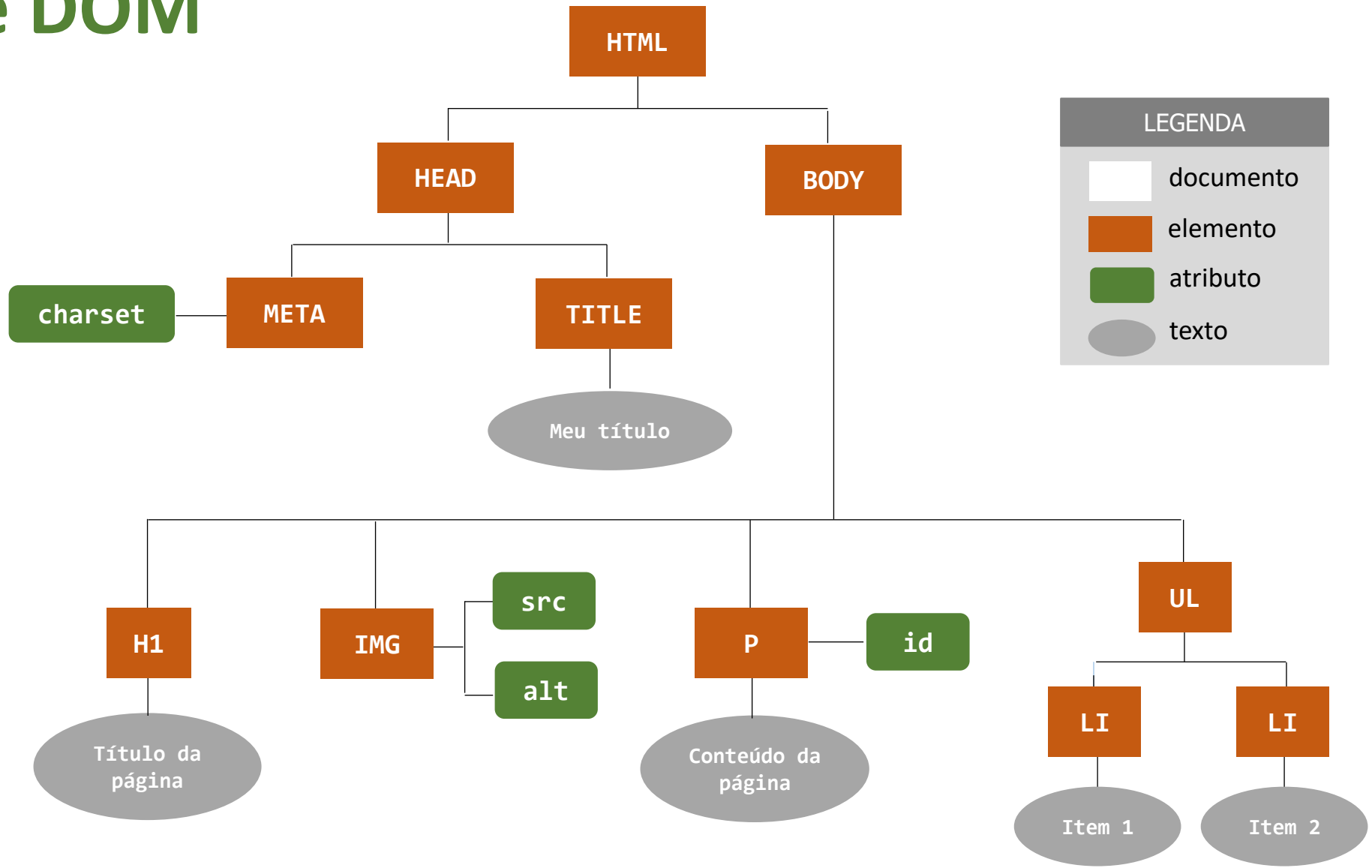
DOM HTML

> Representação da **estrutura do documento HTML**

> Diagrama representativo do **tipo árvore**

> Família com **graus de parentesco**

Árvore DOM



O DOM HTML é um padrão para acessar, adicionar, modificar ou remover elementos e atributos HTML.

Selecionando elementos

Tipos de seleção de elementos possíveis

- `tagName`
- `ID`
- `name`
- `className`
- `seletor`

Uma das coisas mais úteis para realizar utilizando o DOM, **é selecionar elementos da árvore**, e assim **manipular esses elementos**.

Existem **diversos métodos** que podem ser utilizados para esta finalidade.

Seleção de elementos por tag

getElementsByTagName("tag")

Método que retorna os elementos de uma determinada **tag**.

- **tagName**
- ID
- name
- className
- seletor

```
...
</head>
  <body>
    <h1>Iniciando com o DOM!</h1>

    <p>Primeiro parágrafo</p>

    <p>Segundo parágrafo</p>

    <script>
       var paragrafo1 = document.getElementsByTagName('p')[0];

        document.writeln("O que temos no primeiro P? " + paragrafo1.innerText);

    </script>

  </body>
</html>
```

< tag >

Retorna um conjunto de elementos, de determinada tag.

getElementsByTagName("tag")

Acessa **todos os elementos do DOM do tipo definido** no parâmetro.

Retorna uma **coleção de objetos** (elementos HTML) na **mesma ordem da marcação**.

```
<!DOCTYPE html>
  <html>
     <head>
        <script>
              window.onload = function(){
                  var elementos = document.getElementsByTagName("p");

                  elementos[0].style.backgroundColor = "#f00";
                  elementos[1].style.backgroundColor = "#0f0";
                  elementos[2].style.backgroundColor = "#00f";
              }
        </script>
     </head>
     <body>
         <p>Texto do primeiro parágrafo</p>
         <p>Texto do segundo parágrafo</p>
         <p>Texto do terceiro parágrafo</p>
     </body>
</html>
```

A variável **elementos** é uma coleção contendo três parágrafos.

Pode-se acessar cada **elemento individualmente** através de um **índice numérico**.

getElementsByTagName("tag")


```
<!DOCTYPE html>
  <html>
     <head>
        <script>
            window.onload = function(){
              var i, elementos = document.getElementsByTagName("p");

                for(i = 0; i < elementos.length; i++){
                  elementos[i].style.backgroundColor = "#f00";
                }
            }
        </script>
     </head>
     <body>
         <p>Texto do primeiro parágrafo</p>
         <p>Texto do segundo parágrafo</p>
         <p>Texto do terceiro parágrafo</p>
     </body>
</html>
```

A propriedade **length** acessa o tamanho da coleção, ou seja, o número total de elementos.

Pode-se usar uma **estrutura de repetição** para iterar entre os elementos.

Seleção de elementos por ID

- tagName
- **ID**
- name
- className
- seletor

`getElementById("id")`

Retorna um elemento que possui um **`identificador`** específico.

Retorna nulo se nenhum elemento com o ID especificado existir.

Um ID deve ser único em uma página. Se houver mais de um elemento com o ID especificado, o **método retornará o primeiro elemento no código-fonte**.

getElementById("id")


```
<!DOCTYPE html>
  <html>
     <head>
        <script>
           function acessaElemento(){
               var elemento = document.getElementById("noticia");
               window.alert(elemento);
           }
           acessaElemento();
        </script>
     </head>
     <body>
         <p id="noticia">Texto da notícia</p>
     </body>
</html>
```

O método **getElementById()** acessa o elemento do DOM cujo atributo **id** foi definido como parâmetro.

PROBLEMA?

A função é chamada antes do carregamento da página.

Como a página ainda não foi totalmente carregada, teremos um erro, uma vez que o navegador ainda não sabe quem possui este ID.

getElementById("id")


```
<!DOCTYPE html>
  <html>
     <head>
        <script>
           function acessaElemento(){
               var elemento = document.getElementById("noticia");
               window.alert(elemento);
           }
        </script>
     </head>
     <body onload="acessaElemento();">
         <p id="noticia">Texto da notícia</p>
     </body>
</html>
```

Fix para o problema: associar a execução do script a um evento.

A função **acessaElemento()** é disparada a partir do evento **onload**, ou seja, após o carregamento da página.

getElementById("id")

O evento **onload** pode disparar uma função anônima diretamente no JS. Pode ser útil para disparar funções após o carregamento da página.

```
<!DOCTYPE html>
  <html>
     <head>
        <script>
             window.onload = function(){
                var elemento = document.getElementById("noticia");
                alert(elemento);
            }
        </script>
     </head>
     <body>
         <p id="noticia">texto texto</p>
     </body>
</html>
```

Propriedade innerHTML

Permite o **acesso** ou a **definição** do conteúdo HTML de um elemento do DOM.

```
<!DOCTYPE html>
 <html>
    <head>
       <script>
          window.onload = function(){
             var elemento = document.getElementById("noticia");
             // Acessando o conteúdo presente no parágrafo notícia
             alert(elemento.innerHTML);

             // Alterando conteúdo no parágrafo
             elemento.innerHTML = "Novo texto modificado";
          }
       </script>
    </head>
    <body>
        <p id="noticia">Texto da notícia</p>
    </body>
</html>
```

Este **conteúdo HTML** pode ser simplesmente texto ou uma estrutura mais complexa, contendo mais elementos de marcação com textos.

Propriedade innerHTML

Associando a ação a um evento

```
<!DOCTYPE html>
 <html>
    <head>
       <script>
          window.onload = function(){
             var elemento = document.getElementById("conteudo");
             var botao = document.getElementById("btn");

             botao.onclick = function(){
                elemento.innerHTML = "<p>Novo texto</p>";
             }
          }
       </script>
    </head>
    <body>
       <div id="conteudo"></div>
       <button id="btn">Adicione o texto!</button>
    </body>
</html>
```

Atrelando o evento **onclick** ao botão. O novo parágrafo só será inserido dentro da **div** após o clique no botão.

Propriedade `element.style`

A propriedade **`style`** permite a definição de **regras CSS** em elementos HTML através do JavaScript.

As **propriedades CSS compostas de duas palavras separadas por hífen** devem ser escritas em ***camelCase*.**

CSS	JavaScript
`background-color`	`backgroundColor`
`box-shadow`	`boxShadow`
`padding-left`	`paddingLeft`

Lista completa de propriedades: http://www.w3schools.com/jsref/dom_obj_style.asp

Propriedade element.style

```
<!DOCTYPE html>
 <html>
    <head>
       <script>
           window.onload = function(){
                var elemento = document.getElementById("noticia");
                elemento.style.backgroundColor = "#f00";
                elemento.style.border = "1px solid #000";
                elemento.style.fontFamily = "Tahoma";
                elemento.style.color = "#fff";
           }
       </script>
    </head>
    <body>
        <p id="noticia">Texto da notícia</p>
    </body>
</html>
```

Aplicando diversos estilos ao parágrafo **notícia**: cor de fundo, borda, fonte e cor do texto.

Imprimindo variáveis


```
<!DOCTYPE html>
<html>
    <head>
        <title>JavaScript</title>
        <meta charset="utf-8">
        <script>
            var nomeAluno = "Ana Paula";
            document.getElementById("bloco").innerHTML = nomeAluno;
        </script>
    </head>
    <body>
        <p id="bloco"></p>
    </body>
</html>
```

O conteúdo da variável **nomeAluno** é inserido no elemento com a ID **bloco** através do método **document.getElementById()** por meio da propriedade **innerHTML**.

getElementsByName("name")

- tagName
- ID
- **name**
- className
- seletor

O objeto **HTMLCollection** representa uma coleção de nodos. Os nodos podem ser acessados por números de índice. **O índice começa em 0**.

getElementsByName("name")

Método que retorna uma coleção com todos os elementos (objeto HTMLCollection) que possuem uma determinada propriedade **name**.

No HTML5, o atributo "name" está obsoleto. e foi substituído pelo atributo "id" para muitos elementos.

name x id

Atributo name

O atributo **name** deve ser usado principalmente para identificar campos de formulários. Sempre que realizamos um "**submit**" de um formulário, o atributo **NAME** é o identificador dentro de uma requisição **GET** ou **POST** no servidor.

Atributo id

O atributo **id** é geralmente usado para facilitar a identificação de elementos que vão receber um estilo **CSS** ou até mesmo uma função **Javascript**.

getElementsByName("name")


```
<body>

    EC Pelotas:       <input name="clube" type="checkbox" value="Pelotas">
    GE Brasil:        <input name="clube" type="checkbox" value="Brasil">
    SC Internacional: <input name="clube" type="checkbox" value="Internacional">
    Grêmio FBPA:      <input name="clube" type="checkbox" value="Grêmio">

    <p>Clique aqui para selecionar todos os elementos que tem atributo name igual a "clube".</p>

    <button onclick="checkAll()">Selecione todos</button>

    <script>
        function checkAll() {
            var i, x = document.getElementsByName("clube");
            for (i = 0; i < x.length; i++) {
                if (x[i].type == "checkbox") {
                    x[i].checked = true;
                }
            }
        }
    </script>

</body>
```

Busca todos elementos que possuem o atributo **name** igual a **clube.**

getElementsByClassName("class")

- tagName
- ID
- name
- **className**
- seletor

O objeto **HTMLCollection** representa uma coleção de nodos. Os nodos podem ser acessados por números de índice. **O índice começa em 0**.

getElementsByClassName("class")

Método que retorna uma coleção com todos os elementos (**HTMLCollection**) que são de uma **classe** específica.

getElementsByClassName(“class”)


```
<body>
    <div class="exemplo">
        <p>P na primeira div com class="exemplo". Índice da div é 0.</p>
    </div>

    <div class="exemplo cor">
        <p>P na primeira div com class="exemplo cor". Índice da div é 0.</p>
    </div>

    <div class="exemplo cor">
        <p>P na segunda div com class="exemplo cor". Índice da div é 1.</p>
    </div>

    <p>Clique para alterar o estilo da primera div com as classes "exemplo" e “cor".
    </p>

    <button onclick="alteraFundo()">Clique aqui</button>
</body>
```

```
<script>
    function alteraFundo() {
        var x = document.getElementsByClassName("exemplo cor");
        x[0].style.backgroundColor = "darkgreen";
        x[0].style.color = "white";
    }
</script>
```

getElementsByClassName("class")


```
<body>

    <div class="exemplo">
        Uma div com class="exemplo"
    </div>

    <div class="exemplo">
        Outra div com class="exemplo"
    </div>

    <p class="exemplo">Elemento p com class="exemplo".</p>

    <p>Um elemento <span class="exemplo">span</span> com class="exemplo" dentro de outro elemento p.</p>

    <p>Clique para mudar o estilo de todos elementos com class="exemplo".</p>

    <button class="exemplo" onclick="alteraFundo()">Testando</button>

</body>
```

```
<script>
    function alteraFundo() {
        var x = document.getElementsByClassName("exemplo");
        var i;
        for (i = 0; i < x.length; i++) {
            x[i].style.backgroundColor = "darkgreen";
            x[i].style.color = "white";
        }
    }
</script>
```

querySelector("selector")

- tagName
- ID
- name
- className
- seletor

O método `querySelector("selector")` retorna o **primeiro elemento que corresponde ao seletor** especificado no documento.

Para retornar **todas as correspondências**, use o método `querySelectorAll()`

`querySelector("selector")`

`querySelectorAll("selector")`

Forma mais recente de selecionar elementos.

querySelector(“selector”)

Sintaxe: **document.querySelector(*CSS selectors*)**

Especifica seletores CSS para corresponder ao elemento. Usados para selecionar elementos HTML com base em seus **id, classes, tipos, atributos, valores de atributos, etc.**

```
document.querySelector("p");
```

Busca o primeiro elemento `<p>` no documento

```
document.querySelector("p.exemplo");
```

Busca o primeiro elemento `<p>` no documento com `class=“exemplo"`

```
document.querySelector("#teste").innerHTML = “Oi TSI!";
```

Altere o texto do elemento com `id=“teste"`

```
document.querySelector("div > p");
```

Busca o primeiro elemento `<p>` no documento onde o pai é um elemento `<div>`

```
document.querySelector("a[target]");
```

Obtenha o primeiro elemento `<a>` no documento que tem um atributo `target`

querySelector(“selector”)


```
<html>
<head>
    <style>
        a[target] {
            background-color: yellow;
        }
    </style>
</head>

<body>

    <p>O seletor a[target] garante que todos links com um atributo target tenham um fundo amarelo.</p>

    <a href="https://www.ifsul.edu.br">IFSul</a>
    <a href="https://portal.cin.ufpe.br/" target="_blank">CIn - UFPE</a>
    <a href="https://wp.ufpel.edu.br/computacao/ppgc/" target="_top">PPGC - UFPel</a>
    <a href="https://www.ua.pt/" target="_top">Universidade de Aveiro</a>

    <button onclick="addBorda()">Clique</button>

    <script>
        function addBorda() {

            document.querySelector("a[target]").style.border = "10px solid blue";
        }
    </script>

</body>
</html>
```

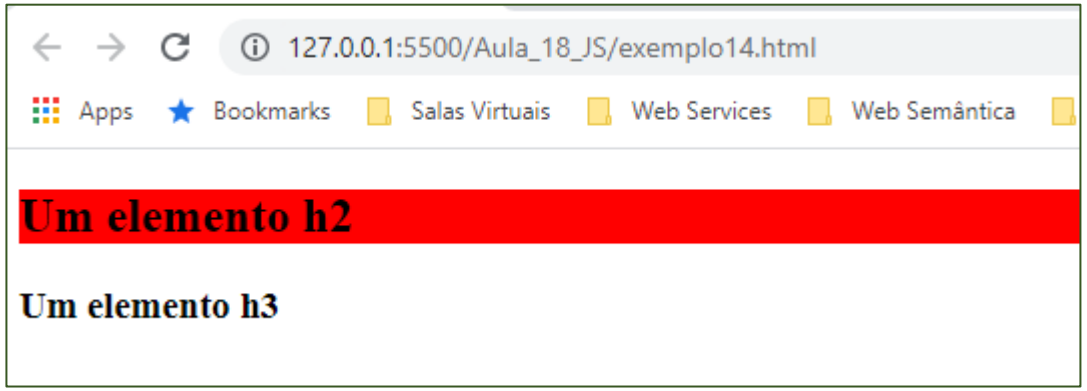
Adiciona uma borda azul ao primeiro `link` com um atributo `target`.

querySelector("selector")

Trabalhando com diversos seletores.

```
<html>

<body>

    <h2>Um elemento h2</h2>
    <h3>Um elemento h3</h3>

    <script>

        document.querySelector("h2, h3").style.backgroundColor = "red";

    </script>

</body>

</html>
```

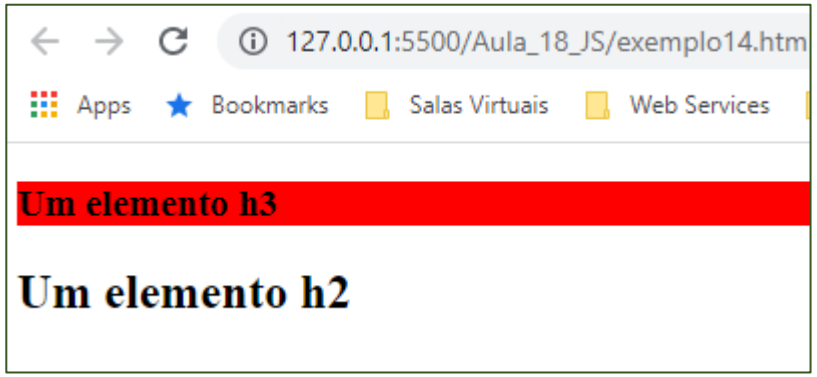
Esse método só retorno o PRIMEIRO elemento que casa com o seletor CSS que especificamos.



querySelector("selector")

E se invertermos a ordem dos elementos?

```
<html>

<body>

    <h3>Um elemento h3</h3>
    <h2>Um elemento h2</h2>

    <script>

        document.querySelector("h2, h3").style.backgroundColor = "red";

    </script>

</body>

</html>
```



querySelectorAll(“selector”)

Método que retorna todos os elementos que correspondem ao seletor CSS especificado.

O objeto retornado é um **NodeList** que representa uma **coleção de nós**, os quais podem ser acessados por meio de números de índice.

```
var x = document.querySelectorAll("p");
x[0].style.backgroundColor = “blue";
```

Recupera todos os elementos <p>.
Configura a backgroundcolor do primeiro como azul

```
var x = document.querySelectorAll("p.examplo");
x[0].style.backgroundColor = "red";
```

Recupera todos os elementos <p> com a classe exemplo.
Configura a backgroundcolor do primeiro como vermelho

```
var i, x = document.querySelectorAll(".exemplo");
for (i = 0; i < x.length; i++) {
    x[i].style.backgroundColor = “green";
}
```

Declara duas variáveis.
Recupera todos os elementos com a classe exemplo.
Configura a backgroundcolor de todos os elementos como verde

querySelectorAll(“selector”)

Método que retorna todos os elementos que correspondem ao seletor CSS especificado.

O objeto retornado é um **NodeList** que representa uma **coleção de nós**, os quais podem ser acessados por meio de números de índice.

```
var i, x = document.querySelectorAll("a[target]");
for (i = 0; i < x.length; i++) {
  x[i].style.border = "10px solid purple";
}
```

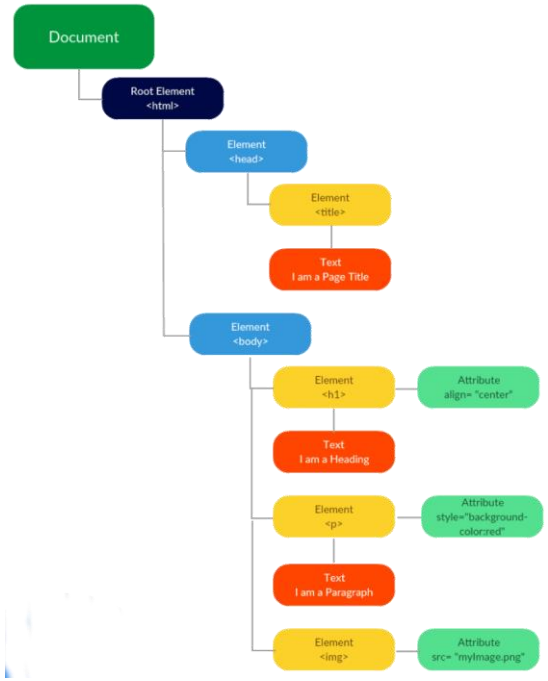
```
var i, x = document.querySelectorAll("div > p");
for (i = 0; i < x.length; i++) {
  x[i].style.backgroundColor = “yellow";
}
```

querySelectorAll(“selector”)


```
<html>
<body>
    <h1>Um elemento H1</h1>
    <h2>Um elemento H2</h2>
    <div>Um elemento DIV</div>
    <p>Um elemento p.</p>

    <p>Um p com um <span style="color:brown;">span</span> dentro.</p>

    <p>Clique para alterar a corde fundo de todos elementos h2, div e span.</p>

    <button onclick="alteraBack()">Clique</button>

    <script>
        function alteraBack() {

            var i, x = document.querySelectorAll("h2, div, span");
            for (i = 0; i < x.length; i++) {
                x[i].style.backgroundColor = "orange";
            }
        }
    </script>

</body>
</html>
```

Configura a cor de fundo como laranja de todos os elementos <h2>, e no documento.

Curso Superior de Tecnologia em Sistemas para Internet
Desenvolvimento de Sites para Web (DSW)
Rafael Cunha Cardoso
rafaelcardoso@ifsul.edu.br



DOM – Document Object Model

INSTITUTO FEDERAL
Sul-rio-grandense
Câmpus Pelotas