Linguagens de Marcação e Scripts

—

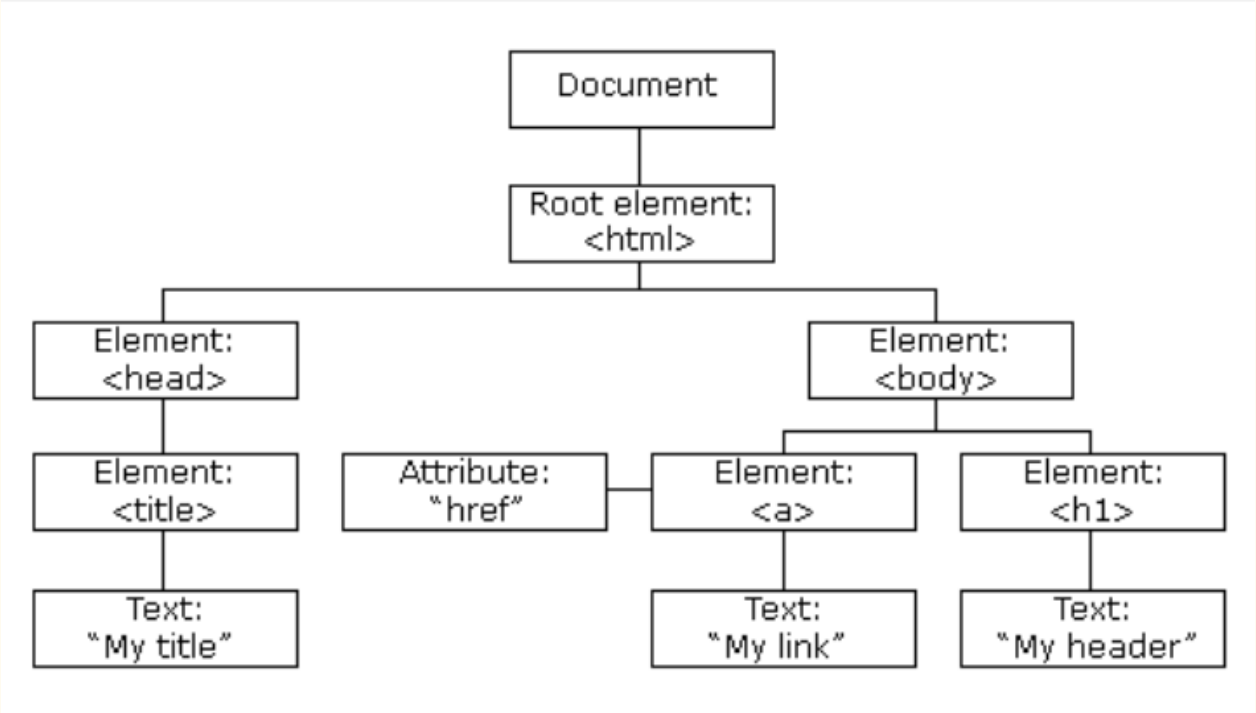
Prof. Aníbal Cavalcante de Oliveira
UFC - QXD0164 - 2019.2

Agenda - Aula 16

- Introdução ao JavaScript para Web.

O Document Object Model do HTML – (HTML DOM)

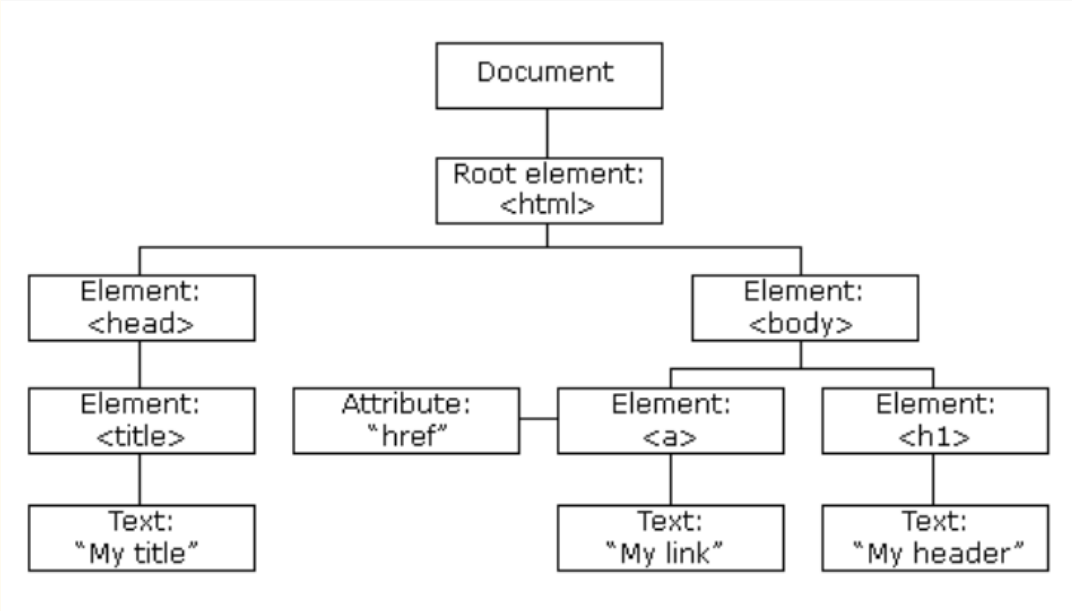
Quando uma página da web é carregada, o navegador cria um **Document Object Model** da página. O modelo DOM HTML é construído como uma árvore de objetos, onde cada nó da árvore é um objeto:



O Document Object Model do HTML – (HTML DOM)

Dessa forma, o código HTML abaixo gera a árvore de objetos DOM. (arquivo: **dom.html**)

```
<!DOCTYPE html>
<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <a href="">My Link</a>
    <h1>My header</h1>
  </body>
</html>
```



O que é o HTML DOM?

O HTML DOM é um padrão para **obter**, **alterar**, **adicionar** ou **excluir** elementos HTML.

No padrão DOM:

1 - Os elementos HTML de uma página são acessados como objetos.

2 - Podemos alterar todas as propriedades desses elementos.

3 - Existem métodos para acessar todos os elementos.

4 - Existem métodos para capturar os eventos de interação do usuário com a página.

5 – Os objetos da página, são representados por nós na árvore DOM, e estão relacionados entre si como: **pais**, **filhos** e **irmãos**.

O objeto "document"

O objeto **"document"** é o objeto pai, ou proprietário de todos os outros objetos em uma página web. Se você deseja acessar qualquer elemento em uma página HTML, você sempre começa acessando o objeto **"document"**.

Abaixo estão alguns exemplos de como você pode usar o objeto de document para acessar e manipular HTML.

```
<script>
   document.getElementById("main"); /*retorna um elemento pelo seu id*/
   document.getElementsByTagName("h1"); /*retorna todos elementos com a tag h1*/
   document.title; /* retorna o título da página*/
   document.body; /* retorna o corpo da página */
   document.images; /* retorna todas as imagens da página*/
   document.links; /* retorna todas as tag que possuem o atributo href*/
</script>
```

Alterando elementos (tags) existentes de uma página HTML através do DOM

A partir do objeto **"document"** podemos modificar elementos existentes de uma página web. Podemos realizar essa tarefa de 3 maneiras:

1 – Adicionar um novo conteúdo a um elemento html.
2 – Alterar o atributo de um elemento html.
3 – Alterar o css de um elemento html.

```
1 -  element.innerHTML -> "Adicionamos um novo conteúdo Html";
2 -  element.attribute -> "Recuperamos o atributo de uma tag";
3 -  elemento.setAttribute -> "Alteramos o atributo de uma tag";
3 -  element.style.property -> "Alteramos um atributo css de uma tag";
```

Alterando elementos - Exemplo 01

Adicionar um novo conteúdo a um elemento html. (**innerhtml01.html**)

```
<p id="demo" onclick="myFunction()">
      Clique nesse parágrafo para modificar seu conteúdo.
</p>

<script>
  var id = 1;
  function myFunction() {
    document.getElementById("demo").innerHTML += "Parágrafo: " + id + "<br>";
    id++;
  }
</script>
```

Alterando elementos - Exemplo 02

Exibir os atributo de uma tag html existente. (**htmlatributes01.html**)

```
<input type="text" name="texto" id="texto" size="50">
<p id="p"></p>

<script>
  var x = document.getElementById("texto");
  var txt = "";
  for (i = 0; i < x.attributes.length; i++) {
    txt = txt + x.attributes[i].name + " = " + x.attributes[i].value + "<br>";
  }
  document.getElementById("p").innerHTML += txt;
</script>
```

Alterando elementos - Exemplo 03

Alterar um dos atributo de uma tag html existente. (**htmlatributes02.html**)

```
<input type="text" name="texto" id="texto" size="20"><br>
<button id-"aumentar" onclick="aumentar()">Aumentar</button>
<button id-"diminuir" onclick="diminuir()">Diminuir</button>
<script>
   var elemento = document.getElementById("texto");
   var atributo = elemento.attributes[3];
   function aumentar(){
      var valor = Number(atributo.value) + 1;
      elemento.setAttribute("size", valor);
   }
   function diminuir(){
      if(atributo.value > 10){
           var valor = Number(atributo.value) - 1;
           elemento.setAttribute("size", valor);
      }
   }
</script>
```

Alterando elementos - Exemplo 03

Alterar o atributo css de uma tag html existente. (**estiloDOM.html**)

```
<p id="p1" onclick="myFunction1()">Hello World!</p>
<p id="p2" onclick="myFunction2()">Hello World!</p>
<p id="p3" onclick="myFunction3()">Hello World!</p>

<script>
          function myFunction1() {document.getElementById("p1").style.color = "blue";}
          function myFunction2() {document.getElementById("p2").style.fontFamily = "Arial";}
          function myFunction3() {document.getElementById("p3").style.fontSize = "larger";}
</script>
```

Adicionar novos, Recuperar e Excluir elementos (tags) através do DOM

Através da propriedade **"document"**, também podemos criar e excluir novos elementos na página web. Cada novo elemento criado é incluído na árvore DOM já existente. Já um elemento excluído será removido da árvore DOM da página.

Métodos para adicionar e excluir nós (tags) na página:

```
1 - document.createElement(element)  -> Cria um novo elemento HTML;
2 - document.createTextNode(texto)   -> Cria um elemento do tipo texto;
3 – elemento.appendChild(element)    -> Adiciona um elemento filho ao elemento;
4 – elemento.removeChild(element)    -> Remove um elemento filho do elemento;
5 - document.replaceChild(new, old)  -> Substitui um elemento por outro.
6 - document.write(text)             -> Escreve um texto na página
7 – element.childNodes               -> Retorna um array com todos o filhos
8 – element.lastElementChild         -> Retorna o último filho elemento
9 – element.firstElementChild        -> Retorna o primeiro filho elemento
10 – element.parentElement           -> Retorna o pai do elemento html
```

Usando document.write e console.log para testar nossos scripts.

As funções **console.log** e **document.write** servem para testar nossos scripts. Cuidado, nunca chame **document.write** após o término do carregamento do documento. Você sobrescreverá o documento inteiro. (**documentWrite.html**)

```
<h2>Minha página da web</h2>

<p>Nunca chame document.write após o término do carregamento do documento. Sobrescreverá o documento inteiro.</p>

<button type="button" onclick="document.write(5 + 6)">Clique Aqui!!</button>

<script>
        var valor = 5 + 6;
        document.write(valor);
        console.log("Olá Mundo: " + valor);
</script>
```

Criar e adicionar um novo elemento na página - Exemplo 01

O método **createElement()** cria um elemento com o nome especificado. Depois de criado, devemos utilizar o método **.appendChild()**. (**createElement01.html**)

```
<p>Clique no botão para criar um elemento P e adicioná-lo na página.</p>
<button onclick="myFunction()">Clique</button>

<script>
    var id = 1;
    function myFunction() {
      var para = document.createElement("p");
      para.innerHTML = "Paragrafo: " + id;
      document.body.appendChild(para);
      id++;
    }
</script>
```

Criar e adicionar um novo elemento na página - Exemplo 02

Nesse exemplo vamos incluir o elemento **P** criado dentro de um elemento div já existente. (**createElement02.html**)

```
<p>Clique no botão para criar um elemento P e adicioná-lo na página.</p>
<button onclick="myFunction()">Clique</button>
<div id="div">Um elemento DIV</div>
<script>
    var id = 1;
    function myFunction() {
      var para = document.createElement("p");
      para.innerHTML = "Paragrafo: " + id;
      document.getElementById("div").appendChild(para);
      id++;
    }
</script>
```

Criar e adicionar um novo elemento em uma lista - Exemplo 03 (createElement03.html)

Nesse exemplo vamos incluir o elemento criado dentro de um elemento ul já existente, mas vamos incluir no começo da lista, com o método **insertBefore().**

```
<input type="texto" name="bebida" id="bebida" value="Água">
<button onclick="myFunction()">Clique</button>
<ul id="lista">
          <li>Café</li>
          <li>Chá</li>
</ul>
<script>
          function myFunction() {
                    var novoItem = document.createElement("li");
                    var texto = document.getElementById("bebida").value;
                    var nodeTexto = document.createTextNode(texto);
                    novoItem.appendChild(noTexto);
                    var lista = document.getElementById("lista");
                    lista.insertBefore(novoItem, lista.lastElementChild);
          }
</script>
```

Remover um elemento na página - Exemplo 04 (removeElement01.html)

Nesse exemplo vamos incluir o elemento criado dentro de um elemento **<ul>** já existente, mas vamos incluir no começo da lista.

```
<ol id="lista">
          <li>Café</li>
          <li>Chá</li>
          <li>Leite</li>
          <li>Água</li>
          <li>Yogurt</li>
</ol>
<button onclick="myFunction()">Clique</button>
<script>
          function myFunction() {
                    var lista = document.getElementById("lista");
                    lista.removeChild(lista.lastElementChild);
          }
</script>
```

Exibir o elemento pai - Exemplo 05

Exibir o elemento pai com o atributo **parentElement** e **nodeName.**

```
<button onclick="myFunction()">Clique Aqui</button>

<p>Exemplo de Lista:</p>
<ul>
          <li id="li">Café</li>
          <li>Chá</li>
          <li>Água</li>
          <li>Leite</li>
          <li>Yorgut</li>
</ul>

<p id="demo"></p>
<script>
          function myFunction() {
                    var x = document.getElementById("li").parentElement.nodeName;
                    document.getElementById("demo").innerHTML = x;
          }
</script>
```

Exibir os elementos filhos - Exemplo 06 (exibirElementosFilhos.html)

A propriedade **childNode** exibe os filhos de um elemento. O espaço em branco dentro dos elementos é considerado como texto e texto é considerado como nós. Comentários também são considerados como nós.

```
<!DOCTYPE html>
<html>
<body><!-- Cometário é um nó -->
    <p>Clique no botão obter informações sobre os nós filhos do elemento body.</p>

    <button onclick="myFunction()">Clique aqui</button>

    <p><strong>Nota:</strong> O espaço em branco dentro dos elementos é considerado como
    texto e texto é considerado como nós. Comentários também são considerados como nós.</p>

    <p id="demo"></p>
    <script>
        function myFunction() {
            var c = document.body.childNodes;
            var txt = "";
            var i;
            for (i = 0; i < c.length; i++) {
                txt = txt + c[i].nodeName + "<br>";
            }
            document.getElementById("demo").innerHTML = txt;
        }
    </script>
</body>
</html>
```

Adicionando um tratador de eventos a um elemento HTML (Reação a eventos)

Podemos adicionar a um elemento HTML uma propriedade para que ele trate eventos do usuário. Em outras palavras, o elemento saberá qual função chamar para um dado evento disparado pelo usuário. Abaixo o botão criado reagirá a vários eventos do usuário.

```
<button id="btn">Clique Aqui</button>
<p id="demo"></p>
<script>
  document.getElementById("btn").onclick = function () {
    document.getElementById("demo").innerHTML = Date();
  }
  document.getElementById("btn").onmouseenter = function () {
    document.getElementById("demo").innerHTML = "Tire o mouse de cima de mim!!"
  }
  document.getElementById("btn").onmouseleave = function () {
    document.getElementById("demo").innerHTML = "Obrigado!!"
  }
</script>
```

Encontrado objetos DOM de uma página.

Existem várias propriedades que retornam objetos e listas de objetos DOM de uma página HTML. A lista completa pode ser vista em: https://www.w3schools.com/js/js_htmldom_document.asp

Property	Description	DOM
document.anchors	Returns all <a> elements that have a name attribute	1
document.applets	Returns all <applet> elements (Deprecated in HTML5)	1
document.baseURI	Returns the absolute base URI of the document	3
document.body	Returns the <body> element	1
document.cookie	Returns the document's cookie	1
document.doctype	Returns the document's doctype	3
document.documentElement	Returns the <html> element	3
document.documentMode	Returns the mode used by the browser	3
document.documentURI	Returns the URI of the document	3
document.domain	Returns the domain name of the document server	1
document.domConfig	Obsolete. Returns the DOM configuration	3
document.embeds	Returns all <embed> elements	3
document.forms	Returns all <form> elements	1
document.head	Returns the <head> element	3
document.images	Returns all <img> elements	1
document.implementation	Returns the DOM implementation	3
document.inputEncoding	Returns the document's encoding (character set)	3
document.lastModified	Returns the date and time the document was updated	3
document.links	Returns all <area> and <a> elements that have a href attribute	1
document.readyState	Returns the (loading) status of the document	3
document.referrer	Returns the URI of the referrer (the linking document)	1
document.scripts	Returns all <script> elements	3
document.strictErrorChecking	Returns if error checking is enforced	3
document.title	Returns the <title> element	1
document.URL	Returns the complete URL of the document	1

Trabalho 02 – Crie uma Lista Aniversário com preenchimento dinâmico em JS.

Crie um formulário para preencher a data de aniversário, o nome, e o e-mail dos seus colegas. Uma tabela com as informações inseridas, deve exibir os dados cadastrados dinamicamente. Todos os campos são obrigatórios. Exemplo: