

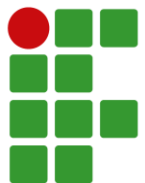


**Licenciatura
em Computação**

Curso de Licenciatura em Computação
Disciplina de Programação I
Professor Tauã M. Cabreira

AULA 15

DOM (Document Object Model)



INSTITUTO FEDERAL
Sul-rio-grandense
Câmpus Pelotas

Variáveis

- Variáveis são utilizadas para armazenar valores.
- JavaScript utiliza a palavra-chave **var** para declarar variáveis.

```
<script>
  // Declaração de variáveis
  var quantProdutos, notaAluno, nomeAluno;
  quantProdutos = 10;
  notaAluno = 8.2;
  nomeAluno = "João";
</script>
```

A notação utilizada na declaração das variáveis é conhecida como **camelCase**.

JS é **case sensitive**, ou seja, difere caracteres maiúsculos e minúsculos.

Variáveis

- JavaScript é uma linguagem de programação não tipificada. Mas as variáveis podem armazenar dados de diferentes tipos: inteiro, ponto-flutuante, string, etc.
- O nome da variável deve começar com: uma letra, um caractere underscore (_), um caractere cifrão (\$)
- **Dica:** Não use o \$ para evitar confusão com códigos específicos de bibliotecas JavaScript.

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 element com a ID **bloco** através do método **document.getElementById()** com a propriedade **innerHTML**.



DOM

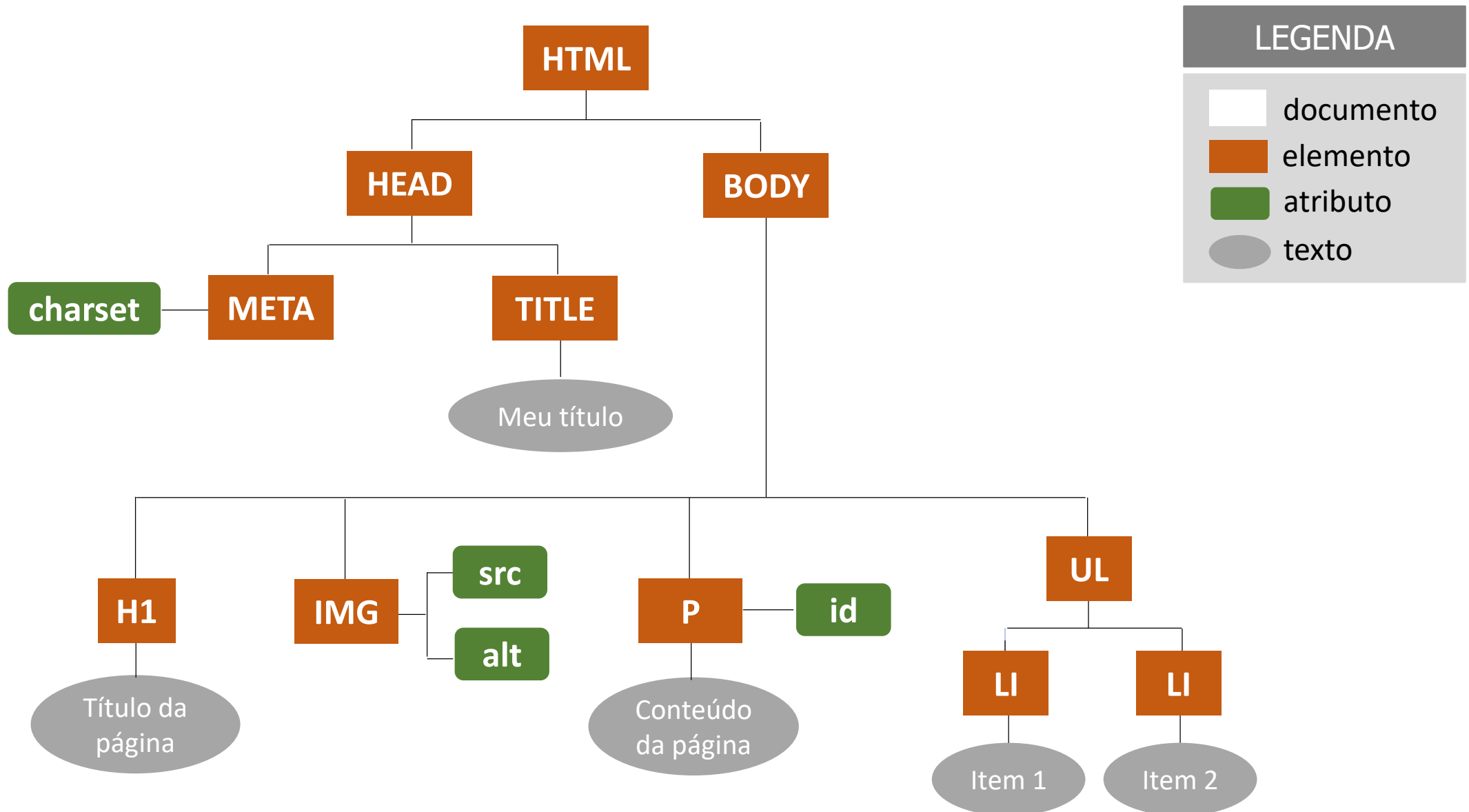
- **Document Object Model** – quando uma página web é carregada, o navegador cria o DOM da página.
- Definição segundo **W3C**:
 - “O DOM – Document Object Model do W3C é uma interface independente de plataforma e linguagem que permite aos programas e scripts acessar e atualizar dinamicamente a **estrutura**, o **conteúdo** e a **estilização** de documentos.”
- Simplifica a tarefa de **acessar** e **manipular** o documento.

DOM

```
<!DOCTYPE html>
<html>
  <head>
    <title>Meu título</title>
    <meta charset="utf-8"/>
  </head>
  <body>
    <h1>Título da página</h1>
    
    <p id="noticia">Conteúdo da página</p>
    <ul>
      <li>Item 1</li>
      <li>Item 2</li>
    </ul>
  </body>
</html>
```

- **DOM HTML:**

- Representação da estrutura do documento HTML
- Diagrama representativo do tipo árvore
- Família com graus de parentesco



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

Objeto Document

Objeto **document**

- O **objeto document** representa um documento (X)HTML ou XML aberto no navegador.
- Este objeto possibilita o acesso via JS a todos os **elementos HTML** de uma página. Sempre comece com **document!**
- Na sequência serão apresentados os seguintes métodos e propriedades: **getElementById()**, **element.style**, **getElementsByTagName** e **innerHTML**.

Método `getElementById()`

```
<!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.



Método `getElementById()`

```
<!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.

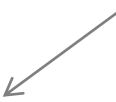


PROBLEMA??

Método `getElementById()`

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


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



Método `getElementById()`

```
<!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>
```

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



Propriedade **element.style**

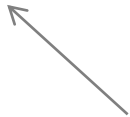
- A propriedade **style** permite a definição de **regras CSS** em elementos HTML através do JS.
- As propriedades CSS compostas de duas palavras separadas por hífen devem ser escritas em **camelCase**:

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.

Propriedade **element.style**

- Atividade:
 - Crie três botões através do element HTML **<button>**.
 - Atrele o evento **onclick** aos botões, disparando em cada botão uma função JS diferente.
 - Crie três funções JS para estilizar um parágrafo com a ID notícia.
 - É possível realizar esta tarefa com apenas uma função?

getElementsByTagName()

- O método **getElementsByTagName()** acessa todos os elementos do DOM do tipo definido no parâmetro.
- Retorna uma coleção de objetos (elementos HTML) na mesma ordem da marcação.
- Pode-se acessar cada elemento individualmente através de um índice numérico.

getElementsByTagName()

```
<!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>
```



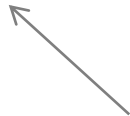
A variável **elementos** é uma coleção contendo três parágrafos.

Através dos índices, pode-se acessar individualmente cada parágrafo e estilizá-lo com uma cor diferente.

getElementsByTagName()

```
<!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.

Propriedade **innerHTML**

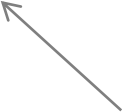
- A propriedade **innerHTML** permite o **acesso** ou a **definição** do conteúdo HTML de um elemento do DOM.
- Este **conteúdo HTML** pode ser simplesmente texto ou uma estrutura mais complexa, contendo mais elementos de marcação com textos.

Propriedade **innerHTML**

```
<!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>
```

Propriedade `innerHTML`

```
<!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.

DOM Nodes

DOM Nodes

- O documento HTML é representado através de uma **árvore**.
- Os elementos HTML são organizados através de uma **hierarquia de nós**. Estes nós estão relacionados como **nós-pais** e **nós-filhos**.
- Os nós-filhos são elementos que encontram-se marcados dentro de outros elementos.
- Pode-se **manipular** a árvore do documento para **criar** e **remover** elementos, seus atributos e os conteúdos.

Criando Novos Elementos

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<script>
```

```
  window.onload = function(){
```

```
    var novoParagrafo = document.createElement("p");
```

```
    var novoTexto = document.createTextNode("Este é novo!");
```

```
    novoParagrafo.appendChild(novoTexto);
```

```
    var elemento = document.getElementById("container");
```

```
    elemento.appendChild(novoParagrafo);
```

```
  }
```

```
</script>
```

```
</head>
```

```
<body>
```

```
  <div id="container">
```

```
    <p id="texto-um">Este é um parágrafo.</p>
```

```
    <p id="texto-dois">Este é outro parágrafo.</p>
```

```
  </div>
```

```
</body>
```

```
</html>
```

O método **createElement** cria um novo elemento HTML vazio.
Já o método **createTextNode** cria um nó de texto.



O método **appendChild** adiciona um nó-filho a um nó-pai.
O texto é adicionado ao parágrafo e o parágrafo é adicionado a
um elemento já existente (DIV container).




Criando Novos Elementos

- O método **appendChild()** adiciona um nó-filho em um nó-pai como último element filho.
- Através do método **insertBefore()** pode-se adicionar um novo nó-filho em qualquer posição dentro de um nó-pai, basta especificar antes de qual nó ele será inserido.

Criando Novos Elementos

```
<!DOCTYPE html>
<html>
  <head>
    <script>
      window.onload = function(){
        var novoParagrafo = document.createElement("p");
        var novoTexto = document.createTextNode("Este é novo!");
        novoParagrafo.appendChild(novoTexto);

        var elemento = document.getElementById("container");
        var primeiro = document.getElementById("texto-um");
        elemento.insertBefore(novoParagrafo, primeiro);
      }
    </script>
  </head>
  <body>
    <div id="container">
      <p id="texto-um">Este é um parágrafo.</p>
      <p id="texto-dois">Este é outro parágrafo.</p>
    </div>
  </body>
</html>
```



O método **insertBefore** adiciona o novo parágrafo antes do parágrafo com a ID texto-um.

Removendo Elementos

```
<!DOCTYPE html>
<html>
  <head>
    <script>
      window.onload = function(){
        var pai = document.getElementById("container");
        var filho = document.getElementById("texto-um");
        pai.removeChild(filho);
      }
    </script>
  </head>
  <body>
    <div id="container">
      <p id="texto-um">Este é um parágrafo.</p>
      <p id="texto-dois">Este é outro parágrafo.</p>
    </div>
  </body>
</html>
```

↖ O método **removeChild** remove um nó-filho de um nó-pai.
É preciso localizar os dois elementos através do método **getElementById**.

Alterando Atributos

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<script>
```

```
    window.onload = function() {
```

```
        document.getElementById("lampada").src = "lampada_on.jpg";
```

```
    }
```

```
</script>
```

```
</head>
```

```
<body>
```

```
    
```

```
</body>
```

```
</html>
```

Pode-se acessar diretamente os atributos de um elemento HTML para modificar os seu valores. Após o carregamento da página, valor do atributo **src** da imagem será alterado para **lampada_on.jpg**



Atividade

- Modifique o exemplo anterior para que a imagem da lâmpada seja modificada apenas a partir do clique de um botão.



Resumo

Método / Propriedade	Finalidade
<code>document.getElementById(id)</code>	Encontra um elemento pelo ID.
<code>document.getElementsByTagName(element)</code>	Encontra elementos pela TAG.
<code>element.innerHTML = novo conteúdo HTML</code>	Modifica o interior de um elemento HTML.
<code>element.attribute = novo valor</code>	Modifica o valor de um atributo de um elemento HTML.
<code>element.style.property = novo estilo CSS</code>	Modifica o estilo de um elemento HTML
<code>document.createTextNode(texto)</code>	Cria um nó do tipo texto.
<code>document.createElement(element)</code>	Cria um nó do tipo elemento.
<code>element.appendChild(element)</code>	Anexa um nó-filho a um nó-pai.
<code>element.insertBefore(element)</code>	Anexa um nó-filho a um nó-pai antes de um elemento.
<code>element.removeChild(element)</code>	Remove um nó-filho de um nó-pai.

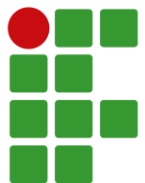


**Licenciatura
em Computação**

Curso de Licenciatura em Computação
Disciplina de Programação I
Professor Tauã M. Cabreira

AULA 15

DOM (Document Object Model)



INSTITUTO FEDERAL
Sul-rio-grandense
Câmpus Pelotas