

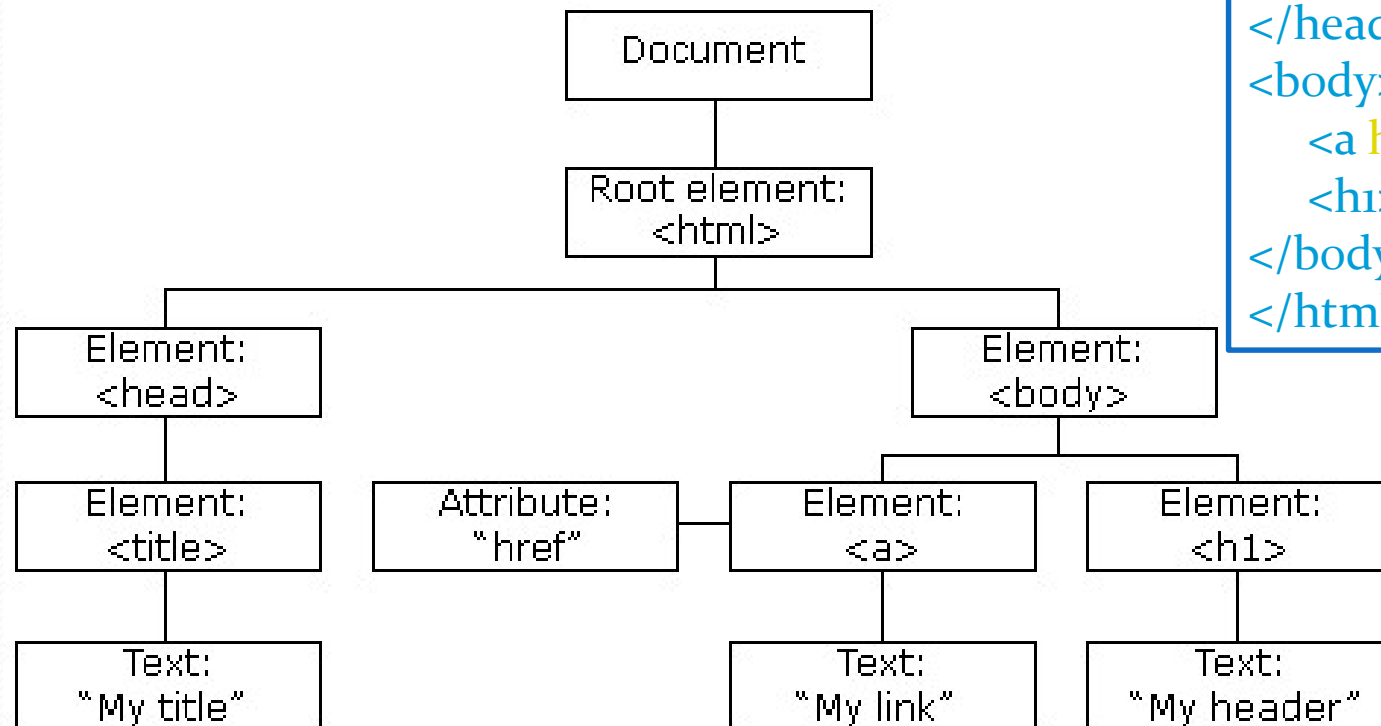
Introdução ao JavaScript – parte3

Profª Mª Denilce Veloso
denilce.veloso@fatec.sp.gov.br
denilce@gmail.com

2021

HTML DOM (Document Object Model)

Quando uma página é lida no navegador ele torna-se um object Document.



```
<html>
<head>
<title>My title</title>
</head>
<body>
  <a href="">My link</a>
  <h1>My header</h1>
</body>
</html>
```

HTML DOM (Document Object Model)

Tabela de Node Type

Node type		nodeName returns	nodeValue returns
1	Element	element name	null
2	Attr	attribute name	attribute value
3	Text	#text	content of node

HTML DOM - Nós

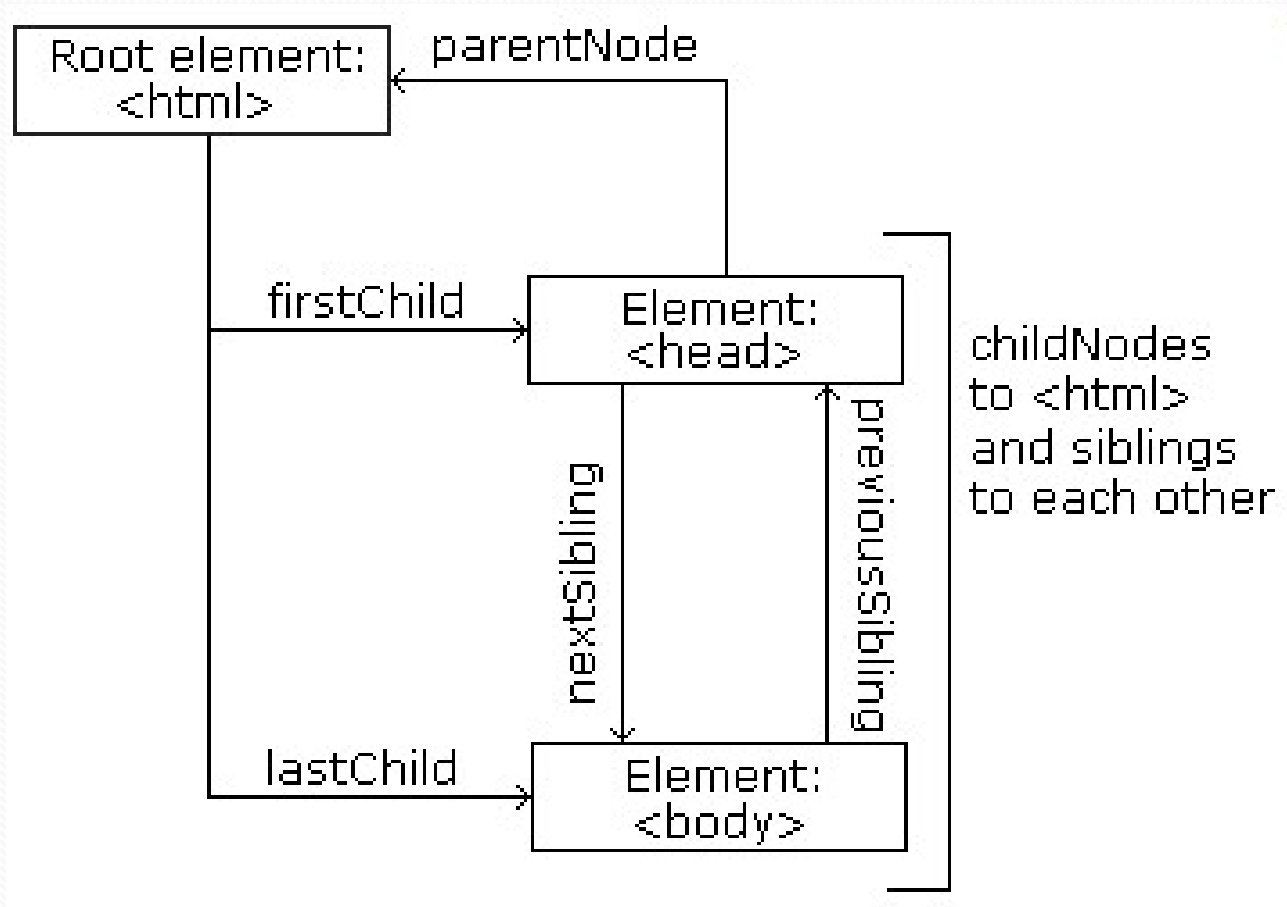
No HTML DOM tudo é um nó.

- ✓ O documento propriamente dito é um nó documento;
- ✓ Todos os elementos HTML são nós de elemento;
- ✓ Todos os atributos HTML são nós de atributo;
- ✓ O texto dentro dos elementos HTML são nós de texto;
- ✓ Os comentários são nós de comentário.

HTML DOM – permite:

- ✓ Encontrar e definir os valores dos elementos em uma página;
- ✓ Manipulação de eventos para os controles em uma página;
- ✓ Modificar os estilos associados aos elementos;
- ✓ Validar e atualizar páginas da web;
- ✓ Etc.

HTML DOM – Relacionamento entre nós



Sibling (“irmão”) – imediatamente anterior ou próximo

HTML DOM – Relacionamento entre nós

```
<html>

<head>
  <title>Exemplo DOM</title>
</head>

<body>
  <h1>Lição Um</h1>
  <p>Alô Mundo!</p>
</body>

</html>
```

Nesse exemplo:

<html> é nó raiz

<html> não tem “parents” ou pais

<html> é parent de <head> e <body>

<head> é o “first child” primeiro filho de <html>

<body> é o “last child” último filho de <html>

<head> e <body> são siblings (irmãos)

<head> tem um “child” filho <title>

<title> tem um filho (text node - texto): “Exemplo DOM “

<body> tem dois filhos <h1> e <p>

<h1> tem um filho (texto) “Lição Um”

<p> tem um filho (texto) “Alô Mundo!”

<h1> e <p> são siblings (irmãos)

<head> e <body> são siblings (irmãos)

HTML DOM – Propriedades

Supondo x (nó) o elemento, atributo ou texto:

- ✓ x.innerHTML: o valor text de x (inclui as tags e textos);
- ✓ x.innerText: o valor text de x (exclui as tags);
- ✓ x.nodeName: o nome do elemento x;
- ✓ x.nodeValue: o valor de x; (por exemplo: do Input)
- ✓ x.nodeType: o tipo de x (1 – elemento; 2 – atributo; 3 – texto);
- ✓ x.parentNode: o nó pai de x;
- ✓ x.childNodes: os nós filhos de x;
- ✓ x.attributes: os nós atributos de x.

HTML DOM - Métodos

Supondo x (nó) o elemento, atributo ou texto:

- ✓ x.getElementById(id): obtém o elemento com o id fornecido;
- ✓ x.getElementsByTagName(name): obtém todos os elementos com a tag name;
- ✓ x.getElementsByClassName(name): obtém todos os elementos com a class name;
- ✓ x.appendChild(node): insere um nó filho node em x;
- ✓ x.removeChild(node): remove o nó filho node de x;
- ✓ x.querySelectorAll(seletor): para encontrar elementos HTML que são iguais a um seletor CSS especificado (id, nome classes, tipos, atributos, valores de atributos, etc). Ex.: `document.querySelectorAll("div.nota, div.alerta");`

HTML DOM – Outras Propriedades e Métodos

- ✓ `hasFocus` – Método retorna `true` se o item possui o foco;
- ✓ `activeElement` – Contém o elemento que possui o foco no momento.
- ✓ `getSelection` – Retorna a seleção atual.

HTML DOM – Ex.: UsandoDOM1GetElementById.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
</head>
<body>
    <p id="primeiro"> Boa Noite Turma 1 </p>
    <input type="text" name="texto1" value="AAA" size=20>
    <br>
    <br>
    ESCOLHA1: <input type="checkbox" name="check" >
    ESCOLHA2: <input type="checkbox" name="check" >
    <script>
        txt = document.getElementById("primeiro").innerHTML;
        document.write("<p>Texto do parágrafo primeiro: " + txt + "</p>");
        var x = document.getElementsByName("check");
        var i;
        for (i = 0; i < x.length; i++) {
            if (x[i].type == "checkbox") {
                x[i].checked = true;
            }
        }
    </script>
</body>
</html>
```

HTML DOM – Exemplo 2 (1)

Ver exemplo UsandoDOM2Nomes.html – como acessar os valores dos campos

```
<!DOCTYPE html>
<html lang="pt-br">

<head>
  <meta charset="UTF-8">
  <title>Teste DOM</title>
  <script>
    function validarDados() {

      // usando a posicao no formulario
      if (document.forms.formulario1.elements[0].value == "" ||
document.forms.formulario1.elements[0].length < 3) {
        alert("Preencha campo NOME corretamente!");
        document.forms.formulario1.elements[0].focus();
        return false;
      }
    }
  </script>
</head>
<body>
```


HTML DOM – Exemplo 2 (2)

Ver exemplo UsandoDOM2Nomes.html – como acessar os valores dos campos

```
// usando o nome no formulario
if (document.forms.formulario1.elements["idEmail"].value == "" ||
document.forms.formulario1.elements["idEmail"].value.indexOf('@') == -1 ||
document.forms.formulario1.elements["idEmail"].value.indexOf('.') == -1) {
    alert("Preencha campo E-MAIL corretamente!");
    document.forms.formulario1.elements.idEmail.focus();
    return false;
}

//usando o nome no formulario de maneira direta
if (document.forms.formulario1.elements.idMensagem.value == "" ||
document.forms.formulario1.elements.idMensagem.value.length < 50) {
    alert("É necessario preencher o campo MENSAGEM com mais de 50
caracteres!");
    document.getElementById("idMensagem").focus();
    return false;
}

return true;
}
</script>
```

HTML DOM – Exemplo 2 (3)

Ver exemplo UsandoDOM2Nomes.html – como acessar os valores dos campos

```
</head>
```

```
<body>
```

```
  <form action="teste" method="POST" name="formulario1" id="formulario1"
onsubmit="return validarDados();">
```

```
    <div id="principal" width=200 height=200 align=center>
```

```
      Nome Completo:
```

```
      <input name="idNome" type="text" id="idNome" size="52"
maxlength="150">
```

```
      <br>
```

```
      <br> E-Mail :
```

```
      <input name="idEmail" type="email" id="idEmail" size="52"
maxlength="150">
```

```
      <br>
```

```
      <br> Mensagem :
```

```
      <textarea name="idMensagem" cols="50" rows="8" id="idMensagem"
value="" input></textarea>
```


HTML DOM – Exemplo 2 (4)

Ver exemplo UsandoDOM2Nomes.html – como acessar os valores dos campos

```
<br>
    <br>
    <br>
    <input name="Submit" type="submit" form="formulario1"
value="Submit">
    <input name="Reset" type="reset" form="formulario1"
value="Redefinir">
    </div>
</form>
</body>

</html>
```

HTML DOM – Exemplo 3 (1)

Ver exemplo UsandoDOM3Nos.html – Trabalhando com os Nós.

```
<!DOCTYPE html>
```

```
<html lang="pt-br">
```

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <title>Teste DOM</title>
```

```
  <script>
```

```
    // adiciona item no nó
```

```
    function adicionalItem(nome) {
```

```
      document.getElementById('Lista').innerHTML += "<li>" + nome + "</li>";
```

```
    };
```

```
    // mostra os dados do nó
```

```
    function mostraNo(item) {
```

```
      alert("Valor innerHTML do principal(Lista)" +
```

```
      document.getElementById(item).innerHTML);
```


HTML DOM – Exemplo 3 (2)

Ver exemplo UsandoDOM3Nos.html – Trabalhando com os Nós.

```
alert("Nome do Nó " + document.getElementById(item).nodeName);
```

```
// se fosse button por exemplo o value seria o texto
```

```
// no caso lista nao tem value
```

```
alert("value do Nó " + document.getElementById(item).nodeValue);
```

```
//pai
```

```
alert("Parent " + document.getElementById(item).parentNode.nodeName);
```

```
// o primeiro filho de elemento Lista é tipo texto
```

```
alert("Primeiro Filho " +
```

```
document.getElementById(item).firstChild.innerHTML + " type " +  
document.getElementById(item).firstChild.nodeType);
```

```
document.getElementById(item).firstChild.nodeValue;
```

```
alert("Ultimo Filho " +
```

```
document.getElementById(item).lastChild.innerHTML + " type " +  
document.getElementById(item).lastChild.nodeType);
```

HTML DOM – Exemplo 3 (3)

Ver exemplo UsandoDOM3Nos.html – Trabalhando com os Nós.

```
    alert("Filho posicao 0 " +  
document.getElementById(item).childNodes[0].innerHTML + " type " +  
document.getElementById(item).childNodes[0].nodeType);
```

```
    alert("Filho posicao 1 " +  
document.getElementById(item).childNodes[1].innerHTML);
```

```
    alert("Próximo sibling " +  
document.getElementById(item).nextSibling.nodeName);  
};  
</script>
```

```
</head>
```


HTML DOM – Exemplo 3 (4)

Ver exemplo UsandoDOM3Nos.html – Trabalhando com os Nós.

```
<body id="principal">
```

```
  <ul id="Lista">
```

```
    <li>Item1</li>
```

```
  </ul>
```

```
  <ul id="Lista2">
```

```
    <li>Item1 da Lista2</li>
```

```
    <li>Item2 da Lista2</li>
```

```
    <li>Item3 da Lista2</li>
```

```
  </ul>
```

```
  <script>
```

```
    adicionalItem("Item2");
```

```
    adicionalItem("Item3");
```

```
    mostraNo("Lista")
```

```
  </script>
```

```
</body>
```

```
</html>
```

PADRÕES NO JAVASCRIPT

Para JS

<https://gist.github.com/vinicius-stutz/1b37cb84b6240efe6ab8137660a15640#:~:text=Segundo%20Dougla s%20Crockford%2C%20todos%20os,como%20o%20C%23%2C%20por %20exemplo.>

<http://crockford.com/javascript/>

https://www.w3bai.com/pt/js/js_conventions.html

Para CSS

<https://desenvolvimentoparaweb.com/css/bem/>
<http://www.andrefelizardo.com.br/blog/padroes-css/#:~:text=trumps-,Padr%C3%B5es%20CSS%2ode%20nomenclatura,CSS%2oescrita%2op ara%2ocada%20p%C3%A1gina.>

ou

<https://wbruno.com.br/css/afinal-como-nomear-ids-e-classes-no-csshtml/>

HTML DOM – Exercício

Qual será o resultado na tela?



```
<p id="teste"> </p>
```

```
<script>
```

```
  var i = 1 + 2 + "3";
```

```
  for (j = 0; j < 10; j++) {
```

```
    if (j === 5) {
```

```
      continue;
```

```
    }
```

```
    document.getElementById("teste").innerHTML += i + j + "<br>";
```

```
  }
```

```
</script>
```

Exercício

✓ Criar um Input Text e dois Input Checkbox.

- 1º CheckBox - transforma o conteúdo do Input Text em letras maiúsculas.
- 2º CheckBox - transforma o conteúdo do Input Text em letras minúsculas.

Dica: *Exemplo UsandoDOM2Nomes.html* mostra exemplos de como acessar os valores dos campos.

Disponibilizar como Atividade11 no GITHUB.

→ Seuusuario/PWEB/Atividade13

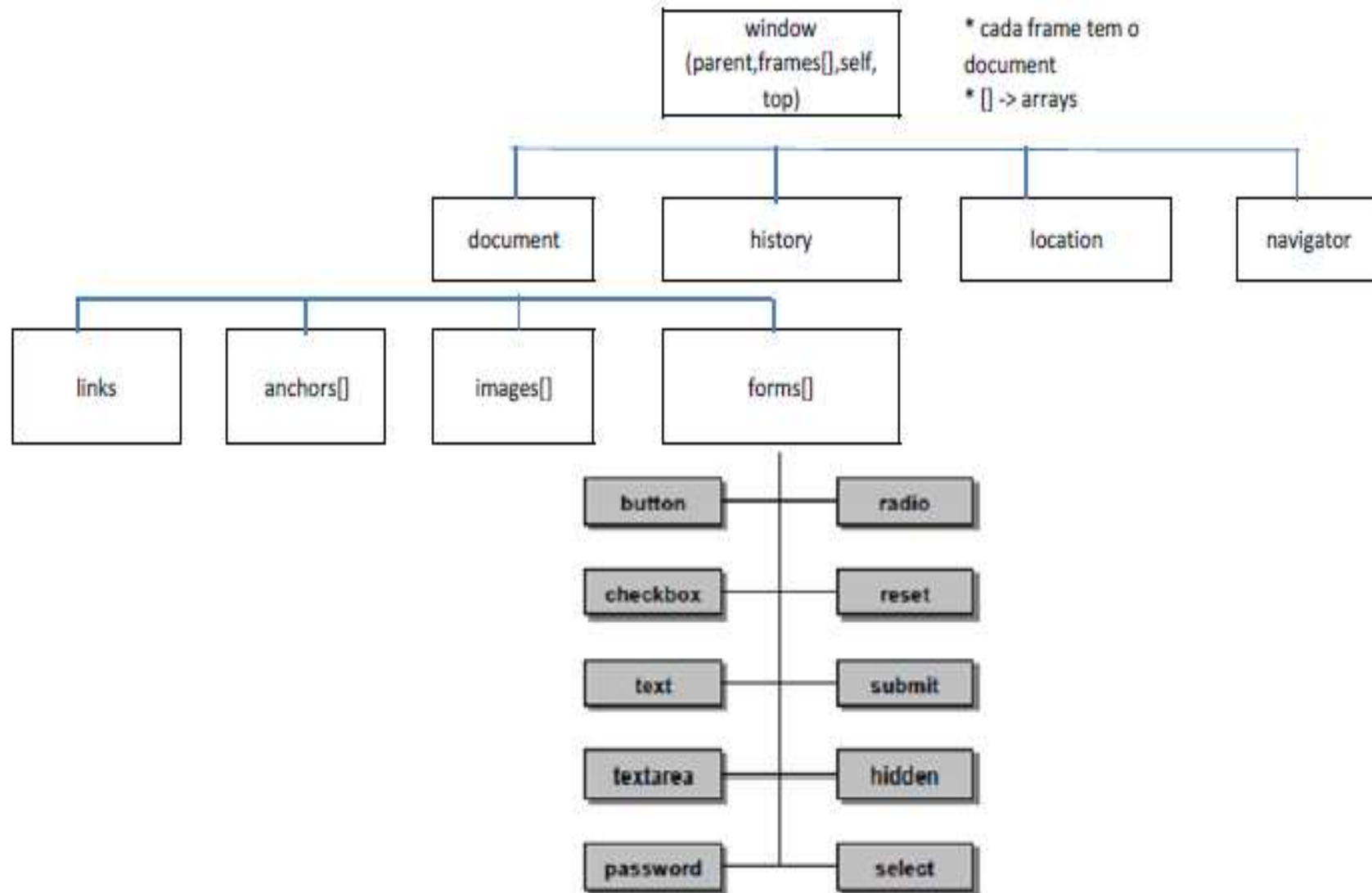
JavaScript – Outros Objetos HTML DOM

- ✓ Para trabalhar com o navegador e os documentos, o JavaScript utiliza objetos browser.
- ✓ Esses objetos estão organizados seguindo uma hierarquia: objeto pai seguido pelo nome ou nomes do objeto filho (separado por pontos).

Exemplo: window.document.imagem1

*** Por convenção, para acessar um objeto, é necessário indicar todo o caminho na hierarquia. Porém, ao escrever document.imagem1, supõe-se estar se referindo a um objeto da janela em uso. Usar caminho completo quando estiver trabalhando com várias janelas.*

JavaScript – Hierarquia objetos DOM



JavaScript – window

- ✓ Window: objeto no topo da hierarquia, representa a janela do navegador, refere-se sempre à janela atual (self).
- ✓ parent - Referência à janela que contém esta janela (só existe quando a janela atual é um frame)
- ✓ top - Referência à janela que não é frame que contém a janela atual (só existe quando a janela atual é um frame)

JavaScript – Objeto Window – Propriedades e Métodos

✓ *window.status* - Altera o conteúdo da linha de *status* do navegador, situada no rodapé da janela.

Ex.:

```
<input type="button" size=20 id="texto" value="passe o mouse em cima"  
onMouseOver="window.status='Descrição da Página 1';return true"  
onMouseOut="window.status="";return true">
```

✓ *window.alert()*

✓ *window.prompt()*

✓ *window.confirm()*

****** *Não precisa usar o nome window antes das propriedades/métodos quando estiver com apenas uma janela aberta.*

JavaScript – Objeto Window – Propriedades e Métodos

- ✓ `window.setTimeout()` - Permite a execução de comandos com temporizador.
- ✓ `window.clearTimeout()` - Interrompe a execução de um temporizador antes do tempo marcado.

EX.:

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8" />
    <title>Objetos DOM</title>
    <script>
        n = 0;

        function atualizar() {
            n++;
            document.getElementById("contador").innerHTML = "<p>" + n + "</p>";
            temp1 = window.setTimeout("atualizar();", 1000);
        }
    </script>
</head>
<body>
    <br><br>
    <span id="contador"></span>

    <a href="#" onClick="window.clearTimeout(temp1);"> Parar o contador
    </a>

    <script>
        atualizar();
    </script>

</body>
</html>
```

****UsandoDom7SetTimeOut.html**

JavaScript – Objeto Window – Propriedades e Métodos

- ✓ *window.open()* - Abre uma nova janela.
- ✓ *window.close()* – Fecha a janela.
- ✓ *window.print()* – Imprime a página.
- ✓ *window.navigator* - Esta propriedade contém informações sobre o navegador, nome, versão, e outras informações. Possui o método:
 - ✓ **javaEnabled()**: informa se o navegador está com o Java habilitado.

JavaScript – Objeto Window – Exemplo: open, close, print

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8" />
  <title>Objetos DOM</title>
```

** Usando Dom8OpenClose.html

Não deixar
bloqueador pop-
ups ativado

```
<script>
  function abreJanela() {
    // nome janela, url, propriedade name, medidas
    novaJanela = window.open("pagina1.html", "janela1", "width = 400, height = 400");
  }
</script>
```

```
</head>
<body>
  <p id="teste"></p>
  <script>
    abreJanela();
    document.getElementById("teste").innerHTML = "nome:" + window.navigator.appName + "
    versão:" + window.navigator.appVersion;
  </script>
```

```
    <input type="button" value="Fecha Nova Janela" onclick="novaJanela.close();">
    <input type="button" value="Imprime esta página" onclick="!print();">
  </body>
</html>
```

JavaScript – Objeto window.load - Exemplo

```
!DOCTYPE html>
```

```
<html lang="pt-BR">
```

```
**UsandoDom5Windowload.html
```

```
<head>
```

```
  <meta charset="UTF-8" />
```

```
  <title>Vendas – Window </title>
```

```
  <style>
```

```
    .coron {
```

```
      background: red;
```

```
    };
```

```
</style>
```

```
<script>
```

```
  window.onload = function () { // load ocorre no carregamento do objeto
```

```
    // +=2 para pular uma linha e ficar listrado
```

```
    var colorir = document.querySelectorAll('tbody tr');
```

```
    for (var i = 0; i < colorir.length; i += 2)
```

```
      colorir[i].className = 'coron'; // atribui nome classe
```

```
  }
```

```
</script>
```

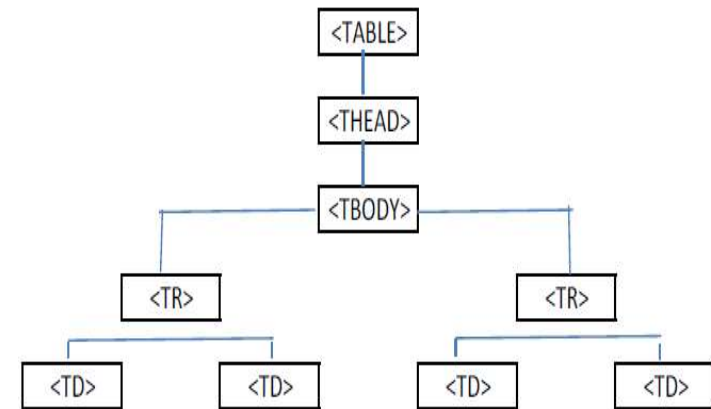
```
</head>
```


JavaScript – Objeto window.load - Exemplo

```
<body>
  <table class="venda">
    <thead>
      <tr> <th>Vendedor</th> <th>Total</th>
    </tr>
    </thead>
    <tbody>
      <tr> <td>Ana</td> <td>10.000,00</td>
      </tr>
      <tr><td>Pedro</td><td>50.000,00</td>
      </tr>
      <tr><td>Maria</td><td>13.200,00</td>
      </tr>
      <tr><td>Celeste</td><td>44.999,00</td>
      </tr>
      <tr><td>Livia</td><td>45.780,00</td>
      </tr>
      <tr><td>Medson</td><td>78.000,00</td>
      </tr>
    </tbody>
  </table>
</body>
</html>
```

→ DENTRO DE UMA LINHA, 2 CABECALHOS

→ TD - CÉLULA



JavaScript – Window Frames

Frames divisões da janela do navegador em múltiplos quadros/painéis. Cada frame pode conter uma página diferente ou a saída de um script. Cada frame é considerado um objeto equivalente ao objeto window (tem os outros objetos, propriedades que o window tem).

JavaScript – Window Frames

```
<!DOCTYPE html>
<html lang="pt-BR">

<head> // declara abaixo do head
<meta charset="UTF-8" />
        <title>WINDOW FRAMES</title>
</head>

        <frameset cols=" * , * , *">
        <frame name="esquerdo"
src="https://www.vestibularfatec.com.br/home/">
        <frame name="direito"
src="http://www.fatecsorocaba.edu.br/">
        </frameset>
<!-- o terceiro frame está vazio -->

<body>
</body>

</html>
```



*** Ver também UsandoDOM9Frames1.html*

JavaScript – Window Frames - Array

É possível referenciar os frames pelo índice, a partir de 0 (zero). No exemplo ANTERIOR `parent.frames[0]` é equivalente ao frame “esquerdo” e `parent.frames[1]` é o frame “direito”.

Exemplo usando linhas e colunas: [UsandoDOM11Frames3.html](#)

JavaScript – location

Location: contém informação sobre a URL corrente

Propriedades:

- ✓ *window.location.href* – href armazena o URL da página atual.
- ✓ *window.location.protocol* - protocol armazena o protocolo da página atual, basicamente http.

Métodos:

- ✓ *window.location.reload()* - reload() recarrega a página atual .
- ✓ *window.location.replace()* – *replace()* substitui a página atual por outra, mas não atualiza o histórico de navegação (não é possível voltar). Ex.:
`location.replace(" https://www.uol.com.br/");`

** UsandoDOM11Location11.html

**Exemplo sobre navegação de frames usando location: UsandoDOM10Frames2.html

JavaScript – history

History: contém histórico das URLs visitadas

Propriedades:

✓ *window.history.length* - length armazena a quantidade de localizações diferentes visitadas.

✓ *window.history.current* - current, assim como a *window.location.href*, contém o endereço da página atual (só que da página que está no histórico).

✓ *window.history.next* - next contém o endereço da próxima página (para onde o usuário foi e depois retornou), podendo recarregá-la novamente através do botão avançar.

✓ *window.history.previous* - previous armazena o endereço da página anterior (de onde o usuário veio), podendo retornar através do botão retornar.

JavaScript – history

Métodos:

✓ *window.history.go()* - *go()* permite a navegação entre as páginas já visitadas. Argumento com valor positivo ex. *go(1)* avança para a próxima página já visitada. (Next)

Argumento com valor negativo ex. *go(-1)* retorna para a página anterior visitada. (Back).

✓ *window.history.back()* - *back()* retorna à página anterior. (Back) Similar *go(-1)*

✓ *window.history.forward()* - *forward()* avança para a próxima página. (Next) Similar *go(0)*

JavaScript – document

Propriedades e Métodos:

- ✓ *window.document.URL* – URL endereço da página atual
- ✓ *window.document.title* - title armazena o título da página, que é exibido na barra de título do navegador.
- ✓ *window.document.referrer*- referrer armazena o endereço da página anterior (usuário estava visualizando anteriormente).
- ✓ *window.document.lastModified* - lastModified armazena a data da última atualização efetuada na página.
- ✓ *window.document.write()* - write imprime texto em um documento. Para imprimir um novo conteúdo, recarregar a página .
- ✓ *window.document.open()* - open utilizado para reescrever um documento primeiramente limpando o conteúdo anterior. É utilizado em novas janelas. (abrir, escrever e depois fechar)
- ✓ *window.document.close()* - close fecha o documento aberto.

**** Ver exemplo UsandoDOM12Document.html**

JavaScript – document.link

Os links contidos em uma página são tratados como objetos. Cada link faz parte do Array `links[]`. O endereço do primeiro link da página criado em HTML tem o índice 0.

Propriedades

```
<body onload="document.write(document.links[0]);">  
<a href="pagina1.html"> Primeira Página</a>  
</body>
```

****** refere-se à página1

✓ `document.links[]` ou `document.links[].href` - href armazena o endereço do link

✓ `document.links.length` - length armazena o número links existentes na página.

****** Ver exemplo *UsandoDOM13Links.html*

JavaScript – document.anchors

As âncoras (links estabelecidos em qualquer parte do documento, que servem como “marcadores”) contidas em uma página também são tratadas como objetos. Cada âncora faz parte de um Array `anchors[]`. A primeira âncora da página, criada via HTML é `document.anchors[0]`.

✓ `window.anchors[].name` - `name` armazena o nome da âncora contida na página.

✓ `window.anchors.length` - `length` armazena o número de âncoras existentes na página:

****** *Ver exemplo [UsandoDOM14Ancoras.html](#)*

JavaScript – document.images

As imagens também são objetos. As imagens do código HTML são armazenadas como elementos de um Array Images. A primeira imagem da página é `document.images[0]` .

- ✓ `window.document.images[].name` - name armazena o nome.
- ✓ `window.document.images[].border` - border armazena o valor da borda.
- ✓ `window.document.images[].complete` - complete armazena os valores true/false, indicando se a imagem já foi carregada ou não.
- ✓ `window.document.images[].height` - height armazena o valor da altura.
- ✓ `window.document.images[].width` – width armazena o valor da largura.
- ✓ `window.document.images[].hspace` - hspace armazena o valor do espaçamento horizontal da imagem .
- ✓ `window.document.images[].vspace` - hspace armazena o valor do espaçamento vertical .
- ✓ `window.document.images[].lowsrc` - lowsrc armazena o endereço da pré-imagem, carregada antes da imagem definitiva.

**** Ver exemplo UsandoDOM15Images.html**

JavaScript – document.images – Efeito RollOvers

Efeito Rollovers - Utilizando o objeto image com os eventos pode-se criar efeitos/animações com imagens. Um deles é o *Rollover*, onde uma imagem é substituída por outra quando se passa o ponteiro do mouse sobre ela.

Ex.:

```
<html>
<head>
  <title>Teste com Link</title>
</head>

<body>
  <a href="#" onmouseover="document.images[0].src='janelaaberta.jpg';"
onmouseout="document.images[0].src='janelafechada.jpg';">
    
  </a>
</body>
</html>
```

** UsandoDOM15Rollovers.html

JavaScript – document.images – Pré-Carregamento

Pré-Carregamento de Imagens - Como as imagens demoram a ser carregadas, principalmente se forem pesadas, é possível otimizar seu carregamento, carregando-as previamente em cache antes da sua exibição.

```
<html>
<head>
  <title>Teste com Link</title>
  <script>
    imgs = new Image();
    imgs[0] = "vistaabrindo.gif";
    imgs[1] = "vistajanela.jpg";
  </script>
</head>
<body>
  <img src="" name="imagem1">
  <br>
  <img src="" name="imagem2">
  <script>
    document.images[0].src = imgs[0];
    document.imagem2.src = imgs[1];
  </script>
</body>
<html>
```

** UsandoDOM16CarregamentoImagens.html

JavaScript – document.images – Animação (1)

O objeto *Image* junto o temporizador *setTimeout()* permite a criação de animações através da exibição de uma sequência de imagens.

```
<html>
```

```
<head>
```

```
    <title>Teste Images - Animação Pernalonga</title>
```

```
    <script>
```

```
        // carrega varias imagens iniciadas por perna
```

```
            function carrega() {
```

```
                img = new Image();
```

```
                for (i = 0; i < 4; i++) {
```

```
                    img[i] = "perna" + (i + 1) + ".gif";
```

```
                }
```

```
            }
```

```
            carrega();
```

```
            i = 0;
```

```
        // exibe imagens carregadas
```

```
            function exhibe() {
```

```
                if (i > 3)
```

```
                    i = 0;
```

```
                    document.images[0].src = img[i];
```

```
                    i++;
```

```
                ** UsandoDOM17Animacao01.html
```


JavaScript – document.images – Animação (2)

```
// chamar novamente depois do timeout
```

```
timer = window.setTimeout("exibe();", 1000);
```

```
}
```

```
function para(){
```

```
    window.clearTimeout(timer);
```

```
}
```

```
</script>
```

```
</head>
```

```
<body bgcolor="brown">
```

```
    
```

```
    <br>
```

```
    <input type="button" value="Iniciar" size=20 onclick="exibe();">
```

```
    <input type="button" value="Parar" size=30 onclick="para();">
```

```
</body>
```

```
</html>
```

**** UsandoDOM17Animacao01.html**

JavaScript – Animação usando animation

*** Ver exemplo UsandoDOM18Animacao2.html*

Quer saber mais sobre animação no JavaScript ?

<http://javascript.info/tutorial/animation>

<https://code.tutsplus.com/pt/tutorials/20-best-css-animations--cms-27561>

JavaScript – document.form

Trata-se do formulário em HTML. Ex.:

```
<form      name="form1"      action="mailto:teste@provedor..com"
method="POST" enctype="text/plain">
```

Propriedades

- ✓ *forms[].name* - Armazena o conteúdo de name (nome)
- ✓ *forms[].action* - Armazena o conteúdo de action. (*para onde será enviado, formulário, e-mail, etc*)
- ✓ *forms[].method* - Armazena o conteúdo de method. (*get, post*)
- ✓ *forms[].target* - Armazena o conteúdo de target (*onde ocorrerá o retorno, nova página, novo frame, etc*)
- ✓ *forms[].encoding* - Armazena o conteúdo de enctype (o que será enviado, se os dados serão codificados)
- ✓ *forms[].length* - Armazena a quantidade de elementos (campos) (somente leitura)

Métodos

- ✓ *forms[].submit()* – Envia o formulário indicado
- ✓ *forms[].reset()* – Limpa o formulário indicado

JavaScript – document.form

Para referenciar o formulário, utiliza-se o Array `forms[]` ou o nome do objeto incluído no atributo “name” da tag `<form>` da HTML.

Ex.:

```
document.forms[0].title = "Mudei o título 1";  
document.forms["Formulario1"].title = "Mudei o título 2";  
document.forms.Formulario1.title = "Mudei o título 3";  
document.Formulario1.title = "Mudei o título 4;"
```


JavaScript – document.form.elements

O *Array* `elements` está subordinado a um formulário. É possível utilizar o nome do objeto, definido através do atributo `name` incluído na tag de cada elemento do formulário (exemplo: `<input>` `<textarea>`,`<button>`)

Propriedades

- ✓ `elements[].name` - Armazena o nome do elemento.
- ✓ `elements[].length` - Armazena o comprimento do elemento.

****** *Ver exemplo UsandoDOM19Elements.html*

JavaScript – Navigator

Apresenta informações do navegador.

Propriedades

- ✓ *navigator.appCodeName* - Nome interno do código do navegador, normalmente Mozilla.
- ✓ *navigator.appName* - Nome do navegador.
- ✓ *navigator.appVersion* - Versão utilizada pelo navegador.
- ✓ *navigator.language* - Idioma. Ex.: "in" para inglês, "pt-br" para português do Brasil.
- ✓ *navigator.platform* - Plataforma do computador utilizado pelo navegador, como "Win32", etc.
- ✓ *navigator.cookieEnabled* - Permite cookies.
- ✓ *navigator.onLine* - Está online.
- ✓ *navigator.userAgent* - Cabeçalho do usuário-agente, uma string que o navegador envia para o servidor Web quando solicita uma página da Web. Inclui informações completas da versão.

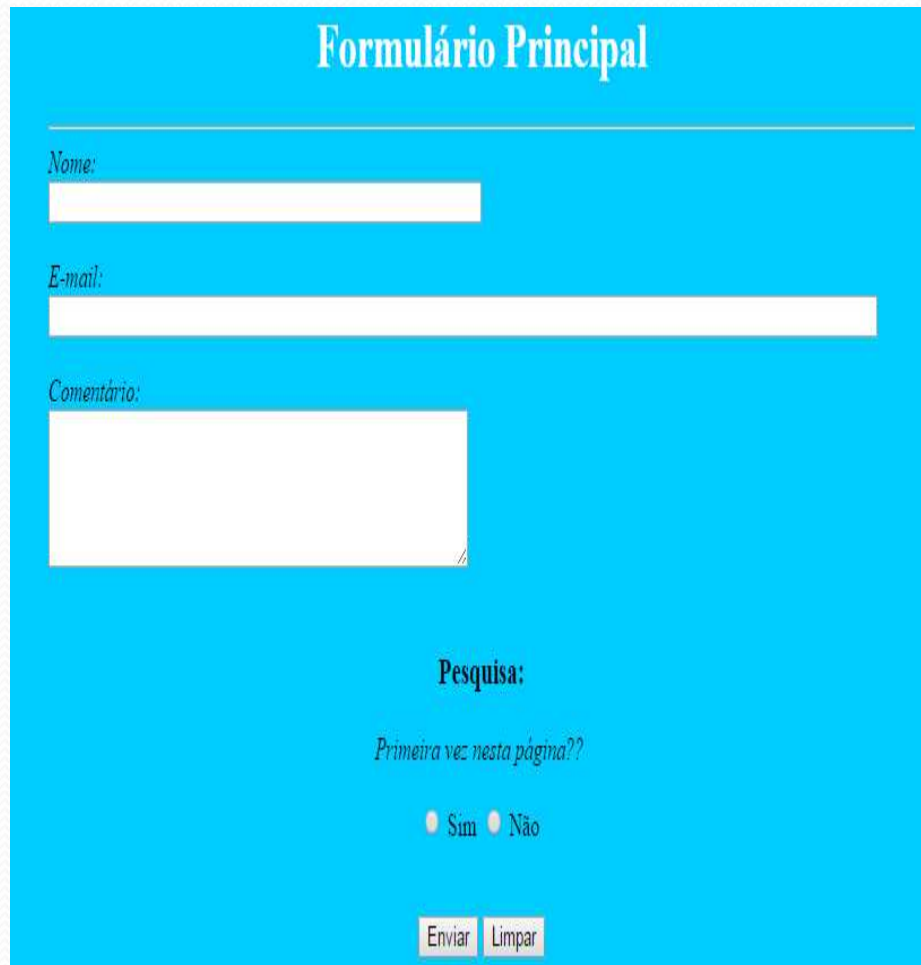
**** Ver exemplo:**

UsandoNavigator1.html

UsandoNavigator2.html

Exercício 1

Criar o seguinte formulário.



The image shows a web form titled "Formulário Principal" on a blue background. It contains the following elements: a text input field for "Nome:", a text input field for "E-mail:", a text area for "Comentário:", a label "Pesquisa:" followed by the text "Primeira vez nesta página??", two radio buttons labeled "Sim" and "Não", and two buttons at the bottom labeled "Enviar" and "Limpar".

- ✓ Criar uma função validar no evento onsubmit do Form.
- ✓ Nome não pode ter menos que 10 caracteres.
- ✓ Email deve ter os caracteres @ e .
- ✓ Comentário deve ter no mínimo 20 caracteres.
- ✓ Pesquisa (obrigatório). Se não: Retornar : “Que bom que você voltou a visitar esta página!”, caso contrário: “Volte sempre à esta página!”. Utilize mesmo atributo name nos radios.
- ✓ Utilizar `document.nomeform.elements[]` na função – pelo menos em algum caso

Disponibilizar como Atividade10 no GITHUB.
→ Seuusuario/PWEB/Atividade14

Exercício 2

Criar uma página utilizando JavaScript que execute as seguintes funções:

- ✓ A página principal deve conter uma caixa de seleção com nomes de cursos (os cursos da Fatec Sorocaba).
- ✓ Quando o usuário escolher um curso, deverá aparecer uma caixa confirmando se a janela contendo o curso deve realmente ser aberta.
- ✓ Caso o usuário confirme (clcando em Ok), o curso escolhido deverá ser carregado em uma nova janela (coloque algumas informações sobre ele) com 600×300 pixels.
- ✓ Use o evento onchange do tag `<select>` para carregar o curso escolhido.

Disponibilizar como Atividade11 no GITHUB.

→ Seuusuario/PWEB/Atividade15

Sobre ES6

Quer conhecer mais sobre o ES6 e outras versões mais recentes?

<https://www.mundojs.com.br/2018/01/27/o-que-e-ecmascript-es6-es8/>

<https://medium.com/@matheusml/o-guia-do-es6-tudo-que-voc%C3%AA-precisa-saber-8c287876325f>

<https://www.treinaweb.com.br/blog/qual-a-diferenca-entre-ecmascript-e-javascript/>

Algumas Diferenças entre JavaScript e TypeScript

<https://www.youtube.com/watch?v=W1RhCMMKrqc>

https://www.alura.com.br/artigos/javascript-ou-typescript?gclid=CjwKCAjwndCKBhAkEiwAgSDKQdPykCacmyOdYHfA4pZu1PskwQWOX7ACYtqghQx6QDN6yMr4NRoCDqYQAvD_BwE

Referências

CAELUM. Apostilas Cursos Gratuitas <https://www.caelum.com.br/apostilas/>. Acesso em: Jan. 2015.

CODEACADEMY. Cursos Gratuitos. <https://www.codecademy.com/pt> Acesso em: Jan. 2015.

SPRITES. Disponível em: <http://www.criarweb.com/artigos/sprites-css3-javascript.html> Acesso em: 24.MAR.2016

DOM. Disponível em: http://www.w3schools.com/jsref/dom_obj_document.asp Acesso em: Jan.2015.

HERANCA. Disponível em: <http://www.devmedia.com.br/classes-no-javascript/23866>. Acesso em: Jan.2015.

HTMMLCSS. Disponível em: <http://del.icio.us/carlosbazilio/{css+html}> Acesso Jan.2015.

JAVASCRIPT_1. Disponível em: <http://www.significados.com.br/javascript/> Acesso Jan.2015.

JAVASCRIPT._2 Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/JavaScript_Vis%C3%A3o_Geral Acesso em: Jan.2015.

OO. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Trabalhando_com_Objeto Acesso em: Jan.2015.

Referências

OPERADORES. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Expressions_and_Operators#Assignment_operators Acesso em: Jan.2016

PEREIRA. Fábio M. Pereira. JavaScript Básico. DESENVOLVIMENTO DE SISTEMAS WEB – 2014.1 UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA CURSO DE CIÊNCIA DA COMPUTAÇÃO. 2014.

PRECEDENCIA. Disponível em: [https://msdn.microsoft.com/pt-br/library/z3ks45k7\(v=vs.94\).aspx](https://msdn.microsoft.com/pt-br/library/z3ks45k7(v=vs.94).aspx) Acesso em: Jan.2015.

SANTOS. Elisabete da Silva. Apostila JavaScript - Faculdade de Tecnologia de São de Paulo.

SILVA. Maurício Samy . JavaScript Guia do Programador. Editora Novatec.

W3SCHOOLS. Disponível em: <http://www.w3schools.com/> Acesso em: Jan.2016.

W3. Disponível em: <http://www.w3.org/> Acesso em: Jan.2016..