JS/DOM/CSS

Prof. Dr. Razer Anthom Nizer Rojas Montaño
2019

Conteúdo

- JavaScript
- DOM
- CSS

JavaScript

JavaScript

- É uma linguagem Script
- Interpretada
- Usada para adicionar interatividade às páginas
- Normalmente, código inserido em páginas
- Gratuita
- NÃO É JAVA

JavaScript

- Linguagem de programação para Designers
- Dinamicamente pode inserir texto no documento
- Pode responder a eventos
- Pode escrever elementos HTML
- Pode ser usado para validar dados de formulários
- Pode detectar dados do cliente, como o navegador
- Manipulação de *cookies*

Escrevendo JavaScript

- Para se colocar código JavaScript

```
<html>
<head><title>Js</title></head>
<body>
  <script type="text/javascript">
      document.write("Hello World!");
  </script>
</body>
</html>
```

Escrevendo JavaScript

- É executado quando chamado
- Seção HEAD:
  - Garante que o *script* será carregado antes de usado
- Seção BODY:
  - *Scripts* executados quando a página é gerada
- Pode-se ter várias seções com *scripts*
- Pode-se incluir um *script* externo:

```
<html>
<head>
  <title>Teste</title>
  <script src="arquivo.js"></script>
</head>
<body>
</body>
</html>
```


Comandos

- Escrever um texto no navegador

```
<script type="text/javascript">
    document.write("Hello World!");
    document.write("<h1>Oi Mundo!!!</h1>");
</script>
```

- Bloco de comandos: { e }

```
<script type="text/javascript">
{
    document.write("Hello World!");
    document.write("<h1>Oi Mundo!!!</h1>");
}
</script>
```

Comentários

- Comentários: `//` ou `/*...*/`

```
<script type="text/javascript">
{
  /*
    Exemplo de bloco de comandos
  */
  document.write("Hello World!"); // Em ingles
  // Em portugues
  document.write("<h1>Oi Mundo!!!</h1>");
  // document.write("<h1>?????</h1>");
}
</script>
```


Variáveis

- Declaração

```
var nome = "Razer";
var idade;
```

- Atribuição

```
idade = 18;
```

- Redeclaração: NÃO perde seu valor anterior

```
var idade;
```

Operações

- Assumindo y = 5

Operador	Descrição	Exemplo	Resultado
+	Adição	x = y + 2;	x = 7
-	Subtração	x = y – 2;	x = 2
*	Multiplicação	x = y * 2;	x = 10
/	Divisão	x = y / 2;	x = 2.5
%	Resto Divisão (módulo)	x = y % 2;	x = 1
++	Incremento	x = ++y; x = y++;	x = 6 x = 5
--	Decremento	x = --y; x = y--;	x = 4 x = 5

Operações

- Assumindo x = 10 e y = 5

Operador	Exemplo	O mesmo que	Resultado
=	x = y;		x = 5
+=	x += y;	x = x + y;	x = 15
-=	x -= y;	x = x – y;	x = 5
*=	x *= y;	x = x * y;	x = 50
/=	x /= y;	x = x / y;	x = 2
%=	x %= y;	x = x % y;	x = 0

Operações

- Concatenação de Strings

```
nome = "Razer";
nome2 = "Anthom";
```

- Operador +

```
nome3 = nome + nome2;
nome3 = nome + " " + nome2;
```

- Ao adicionar, se um operando for String o resultado é sempre String

```
document.write(5 + "5"); // mostra 55
```


Operações

- Operadores Relacionais

Operador	Descrição
==	Igual
===	Exatamente igual
!=	Diferente
<	Menor
<=	Menor ou Igual
>	Maior
>=	Maior ou Igual

Operações

- Operadores Lógicos

Operador	Descrição	Exemplo	Tabela Verdade
!	Não	!a	V → F F → V
&&	E	a && b	V && V → V V && F → F F && V → F F && F → F
||	OU	a || b	V || V → V V || F → V F || V → V F || F → F

Operações

- Operador Ternário

```
res = (hora<19) ? "dia" : "noite";
```

- Equivale a

```
if (hora < 19)
   res = "dia";
else
   res = "noite";
```

Comando if...else

- Comando condicional

```
if (<expressão_booleana>) {
   <comandos>
}

if (<expressão_booleana>) {
    <comandos>
}
else {
    <comandos>
}
```


Comando if...else

- Exemplo

```
<script type="text/javascript">
   var d = new Date();
   var hora = d.getHours();
   if (hora < 19) {
      document.write("Agora é de dia.");
   }
   else {
      document.write("Agora é de noite.");
   }
</script>
```

Comando switch...case

- Avalia a expressão e direciona para o trecho pertinente
- Funciona com *strings*
- Comando condicional

```
switch(<expressão>) {
  case <valor1>:
      Executa se <expressão> == <valor1>
      break;
  case <valor2>:
      Executa se <expressão> == <valor2>
      break;
  ...
  default:
      Executa se <expressão> for diferente de todos
      os casos citados acima
}
```


Comando switch...case

- Exemplo

```
<script type="text/javascript">
   var d = new Date();
   var dia = d.getDay();
   switch (dia) {
      case 5:
        document.write("SEXTA!!!!");
        break;
     case 6:
        document.write("SÁBADOOOO");
        break;
     case 0:
        document.write("Já domingo????");
        break;
     default:
        document.write("Esperando o final de semana");
   }
</script>
```

Janelas de texto

- **Alertas**: mostra uma mensagem com botão OK

```
alert("mensagem");
```

- **Confirmação**: mostra uma pergunta e espera uma resposta: OK/Cancel

```
resp = confirm("mensagem");
if (resp == true) {
...
```

- **Entrada de dados**: mostra uma janela para entrada de um dado string. Pressionando OK retorna o dado entrado. Pressionando CANCEL retorna NULL

```
dado = prompt("Digite seu nome", "Razer");
if (dado!=null)
...
```


Funções

- Palavras reservadas **`function`** e **`return`**
- Função sem parâmetros

```
function mensagem() {
   alert("Mensagem");
}
```

- Função com parâmetros e retorno

```
function soma(x, y) {
   res = x + y;
   return res;
}
```

Funções

- Chamada de função

```
<script type="text/javascript">
   function soma(x, y) {
     res = x + y;
     return res;
   }

   a = 20;
   resultado = soma (a, 50);
   alert(resultado);
</script>
```


Laço for

- Sintaxe como em C

```
for (<início>; <condição>; <incremento>) {
   <comandos>
}
```

- Exemplo

```
for (i=0; i<10; i++) {
   document.write("Número: " + i);
}
```

Laço while

- Sintaxe como em C

```
while (<condição>) {
    <comandos>
}
```

- Exemplo

```
i = 0;
while (i<10) {
  document.write("Número: " + i);
  i++;
}
```


Laço do..while

- Sintaxe como em C

```
do {
    <comandos>
} while (<condição>);
```

- Exemplo

```
i = 0;
do {
  document.write("Número: " + i);
  i++;
} while (i<10);
```

Controle de Fluxo break e continue

- **break**: para a execução de um laço

```
i = 0;
while (i<10) {
   if (i==5)
       break;
   document.write("Número: " + i);
   i++;
}
```


Controle de Fluxo break e continue

- **continue**: para a execução da iteração atual e executa a próxima

```
i = 0;
while (i<10) {
   if (i==5) {
      i = 7;
      continue;
    }
   document.write("Número: " + i);
   i++;
}
```

Laço for..in

- Usado para varrer elementos de um *array*

```
for (<variável> in <objeto>) {
 <comandos>
}
```

- Variável recebe o índice de cada e elemento
- Exemplo:

```
var x;
var arr = new Array();
arr[0] = "Segunda";
arr[1] = "Terça";
arr[2] = "Quarta";
for (x in arr) {
   document.write(arr[x] + "<br />");
}
```


Eventos

- Ações detectadas pelo JavaScript
  - **onLoad**: quando o usuário entra na página
  - **onUnload**: quando o usuário sai da página
  - **onFocus**: quando o controle ganha o foco
  - **onBlur**: quando o controle perde o foco
  - **onChange**: quando o conteúdo do controle é alterado
  - **onMouseOver**: quando o mouse fica sobre o componente
  - **onMouseOut**: quando o mouse sai de cima do componente
  - **onSubmit**: quando um formulário é submetido
- Eventos do DOM (`http://en.wikipedia.org/wiki/DOM_events`)

Eventos

- Exemplos:

```
<form method="post" action="xxx.htm"
    onsubmit="return checkForm()">

<a href="http://www.razer.net.br"
   onmouseover="alert('Mouse por cima');
                                return false">
```


Tratamento de Erros: try..catch

- Uso de **`try..catch`**
  - Tratamento de exceções
  - Erro de sintaxe ou lógica
  - Exceções criadas pelo programador

```
try {
 <comandos>
}
catch(err) {
   <tratamento de erros>
}
```

Tratamento de Erros: throw

- Uso do **`throw`**
  - Levantamento de exceções
  - Pode ser capturado através de try..catch
  - Cria uma nova exceção

```
throw "erro";
```


Tratamento de Erros: Exemplo


```
<script type="text/javascript">
   var x=prompt("Entre um número entre 0 e 10:","");
   try {
      if (x > 10)
            throw "Err1";
      else
            if (x < 0)
                  throw "Err2";
   }
   catch(er) {
      if (er == "Err1")
            alert("Erro! Valor muito alto.");
      if (er == "Err2")
            alert("Erro! Valor muito baixo");
   }
</script>
```

Tratamento de Erros: onerror

- Evento **`onerror`**
  - Função tratadora de erros
  - Se retorna **true**: navegador não mostra erro
  - Se retorna **false**: navegador mostra erro padrão no console

```
onerror=handleErr

function handleErr(msg,url,l) {
  <trata o erro aqui>
  return true/false
}
```


Objetos: String

- Trata textos
- Atributos
  - **`length`**: tamanho do texto
- Métodos
  - **`toUpperCase()`**: retorna em maiúsculo
  - **`toLowerCase()`**: retorna em minúsculo
  - **`indexOf()`**: retorna a posição da primeira ocorrência de uma substring
  - **`substring()`**: retorna uma substring
- Referência:

`http://www.w3schools.com/jsref/jsref_obj_string.asp`

Objetos: String

- Criação

```
var str = new String("teste");
var txt = "oi mundo";
```

- Maiúscula

```
document.write( txt.toUpperCase() );
```

- Separar string

```
var arr = txt.split(" ");
// retorna um array contendo "oi" e "mundo"
```


Objetos: Date

- Trata datas
- Construção

```
var dt = new Date();
```

- Métodos
  - **`getDate()`**: retorna o dia do mês
  - **`getDay()`**: retorna o dia da semana
  - **`getMonth()`**: retorna o mês
  - **`getFullYear()`**: retorna o ano
  - **`getHours()`**: retorna a hora
  - **`getMinutes()`**: retorna os minutos
  - **`getSeconds()`**: retorna os segundos
- Referência:

`http://www.w3schools.com/jsref/jsref_obj_date.asp`

Objetos: Date

- Criação

```
var dt = new Date();
var dt = new Date(<milissegundos>);
var dt = new Date(<data em string>);
var dt = new Date(ano, mês, dia[, hora,
minuto, segundo, milissegundo]);
```

- Mês começa em 0 (zero)
- Formatos das strings
  - **MM-dd-yyyy**
  - **yyyy/MM/dd**
  - **MM/dd/yyyy**
  - **MMMM dd, yyyy**
  - **MMM dd, yyyy**

Objetos: Math

- Operações Matemáticas
- Uso estático

```
var max = Math.max(x, y);
```

- Valores matemáticos
  - **E**: constante de Euler (2.718...)
  - **PI**: constante PI (3.1416...)
  - **SQRT2**: raiz quadrada de 2 (1.414...)
- Métodos
  - **log()**: calcula logaritmo
  - **pow()**: calcula x^y
  - **random()**: retorna um número aleatório entre 0 e 1
  - **round()**: retorna o número arredondado
- Referência:

http://www.w3schools.com/jsref/jsref_obj_math.asp

Objetos: DOM

- Objetos DOM
  - **`Window`**: objeto de mais alto nível da página
  - **`Navigator`**: informações sobre o navegador
  - **`Screen`**: informações sobre a tela (pixels, tamanho, etc)
  - **`History`**: informações páginas visitadas
  - **`Location`**: informações a URL atual

Objetos: DOM

- Objetos DOM
  - **`Document`**: representa o documento HTML
  - **`Anchor`**: representa elemento <a>
  - **`Area`**: representa elemento <area>, dentro de imagens
  - **`Base`**: representa elemento <base>
  - **`Body`**: representa elemento <body>
  - **`Button`**: representa elemento <button>
  - **`Event`**: representa o estado de um elemento
  - **`Form`**: representa elemento <form>
  - **`Frame`**: representa elemento <frame>
  - **`Frameset`**: representa elemento <frameset>

Objetos: DOM

- Objetos DOM
  - **`Iframe`**: representa elemento <iframe>
  - **`Image`**: representa elemento <img>
  - **`Input Button`**: representa um botão de formulário
  - **`Input Checkbox`**: representa um checkbox
  - **`Input File`**: representa um fileupload
  - **`Input Hidden`**: representa um campo hidden
  - **`Input Password`**: representa um campo de senha
  - **`Input Radio`**: representa um radiobutton
  - **`Input Reset`**: representa um botão de reset
  - **`Input Submit`**: representa um botão de submit
  - **`Input Text`**: representa um text box

Objetos: DOM

- Objetos DOM
  - **`Link`**: representa um elemento <link>
  - **`Meta`**: representa um elemento <meta>
  - **`Option`**: representa um option de um select
  - **`Select`**: representa um campo select
  - **`Style`**: representa um estilo
  - **`Table`**: representa um elemento <table>
  - **`TableData`**: representa um elemento <td>
  - **`TableRow`**: representa um elemento <tr>
  - **`TextArea`**: representa um elemento <textarea>

Validação de Formulários

```
<script type="text/javascript">
function validate_form(formulario) {
  if (formulario.email.value == null ||
      formulario.email.value == "") {
        alert("Email é obrigatório.");
        return false; // não submete
  }
  else {
      return true;  // submete
  }
}
</script>
........
   <form action="submitpage.html"  method="post"
          onsubmit="return validate_form(this);">
      Email:<input type="text" name="email" size="30">
      <input type="submit" value="Submit">
   </form>
```


Exercícios

1)Implementar o formulário anterior com a validação de e-mail obrigatório

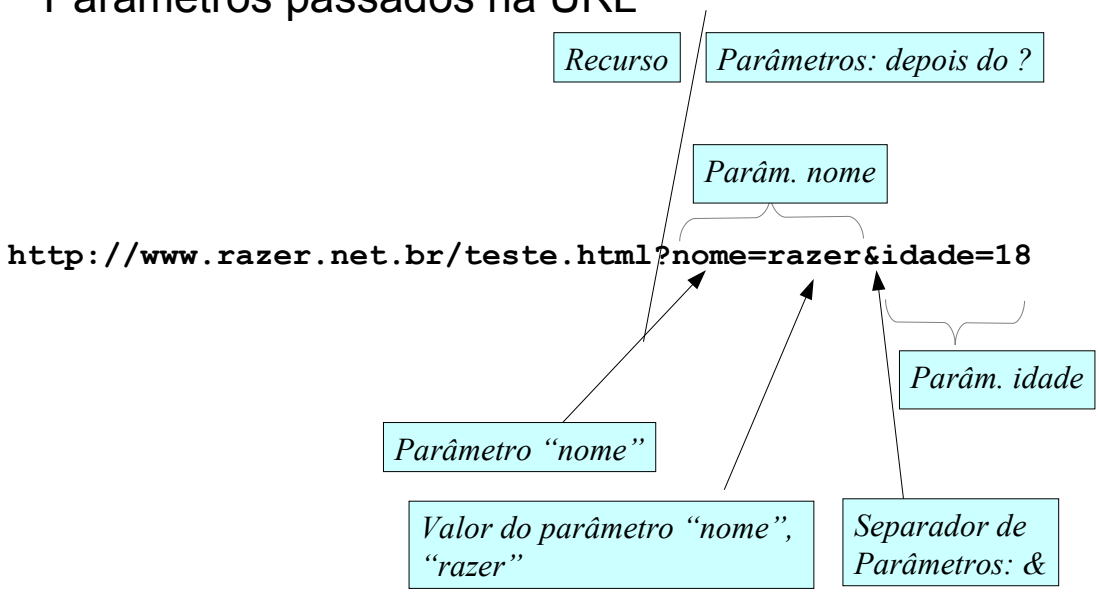
2) Neste mesmo formulário, se o usuário não digitar nada, o componente de texto deve ter sua cor de fundo alterada para vermelho ao perder o foco

3) Neste mesmo formulário, validar e-mail obrigatório e ter mais de 5 caracteres

4) Procurar na internet como validar o formato do e-mail, usando expressões regulares, e implementar

Obtendo Parâmetros GET

- Parâmetros passados na URL



Obtendo Parâmetros GET

```
function queryString(parameter) {
  var loc = location.search.substring(1,
                    location.search.length);
  var param_value = false;
  var params = loc.split("&");
  for (i=0; i<params.length;i++) {
      param_name = params[i].substring(0,
         params[i].indexOf('='));
      if (param_name == parameter) {
          param_value = params[i].substring(
         params[i].indexOf('=')+1)
      }
  }
  if (param_value)
      return param_value;
  else
      return false;
}
```

Obtendo Parâmetros GET

```
function get_url_param(name) {
  name = name.replace(/[\[]/,"\\\[").
            replace(/[\]]/,"\\\]");
  var regexS = "[\\?&]" + name + "=([^&#]*)";
  var regex = new RegExp( regexS );
  var results = regex.exec( window.location.href );
  if( results == null )
      return "";
  else
      return results[1];
}
```


Cookies

- Informações armazenadas no computador de quem acessa um sítio
- Armazenar cookie

```
document.cookie = "var1=valor1;var2=valor2";
```

- Verificar um cookie

```
if (document.cookie.length>0) {
    // precisa fazer parse (; e =)
    c = document.cookie;
}
```

Cookies

- Função para armazenar um cookie

```
function setCookie(c_name, value, expiredays) {
  var exdate = new Date();
  exdate.setDate(exdate.getDate() + expiredays);
  document.cookie = c_name+ "=" + escape(value) +
      ((expiredays==null) ? "" : ";expires=\" +
       exdate.toGMTString());
}
```


Cookies

- Função para verificar um cookie

```
function getCookie(c_name) {
  if (document.cookie.length > 0) {
       c_start=document.cookie.indexOf(c_name + "=");
       if (c_start!=-1) {
             c_start=c_start + c_name.length + 1;
         c_end=document.cookie.indexOf(";",c_start);
            if (c_end == -1)
                    c_end = document.cookie.length;
            return unescape(document.cookie.substring(
                                          c_start,
  c_end));
      }
  }
  return "";
}
```

Cookies..

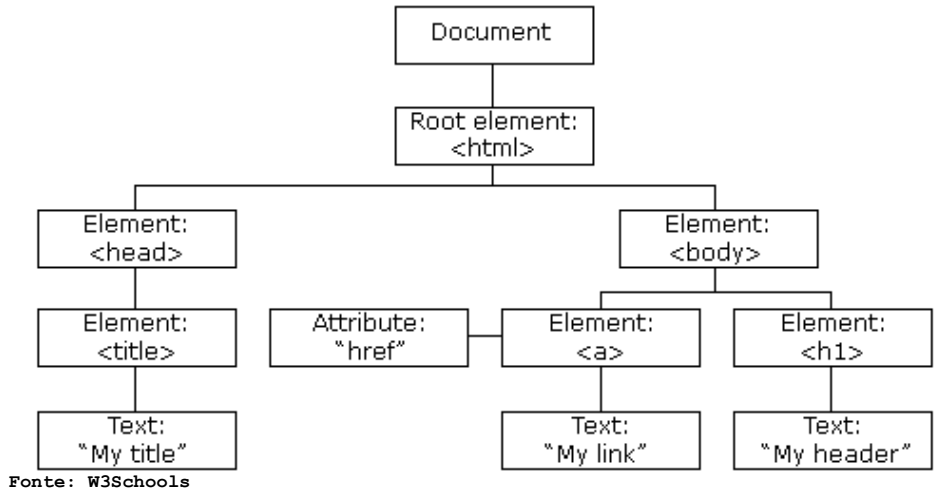
- Exemplo de uso

```
function checkCookie() {
  username=getCookie('username');
  if (username!=null && username!="") {
       alert('Welcome again '+username+'!');
  }
  else {
     username=prompt('Please enter your name:',"");
     if (username!=null && username!="") {
        setCookie('username',username,365);
     }
  }
}
```


DOM: Document Object Model

HTML DOM

- Padrão de acesso aos elementos HTML
- Apresenta o documento HTML como uma árvore



Fonte: W3Schools

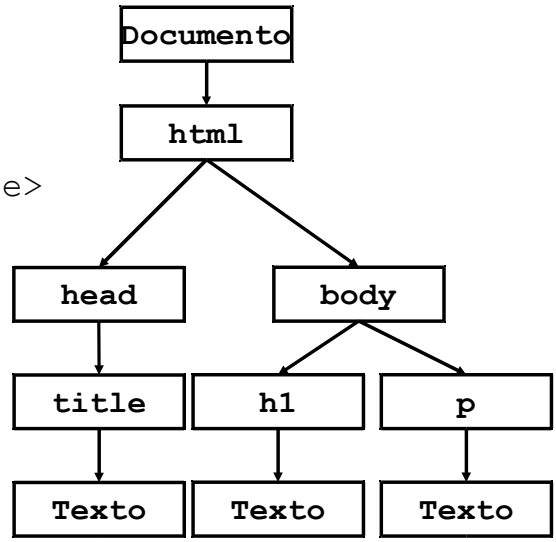
HTML DOM

- Tudo em HTML é um NODO
  - Documento inteiro é um nodo
  - Toda *tag* é um nodo
  - Os textos nos elementos são nodos de texto
  - Todo atributo é um nodo
  - Comentários são nodos de comentários

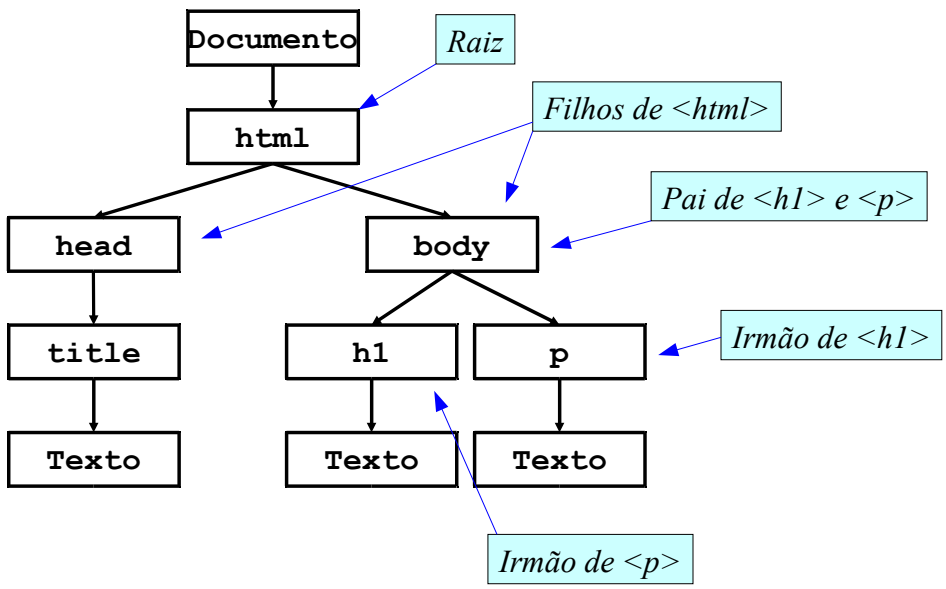
HTML DOM

- Exemplo:

```
<html>
  <head>
     <title>Teste DOM</title>
  </head>
  <body>
     <h1>Teste DOM 1</h1>
     <p>Hello world!</p>
  </body>
</html>
```



HTML DOM

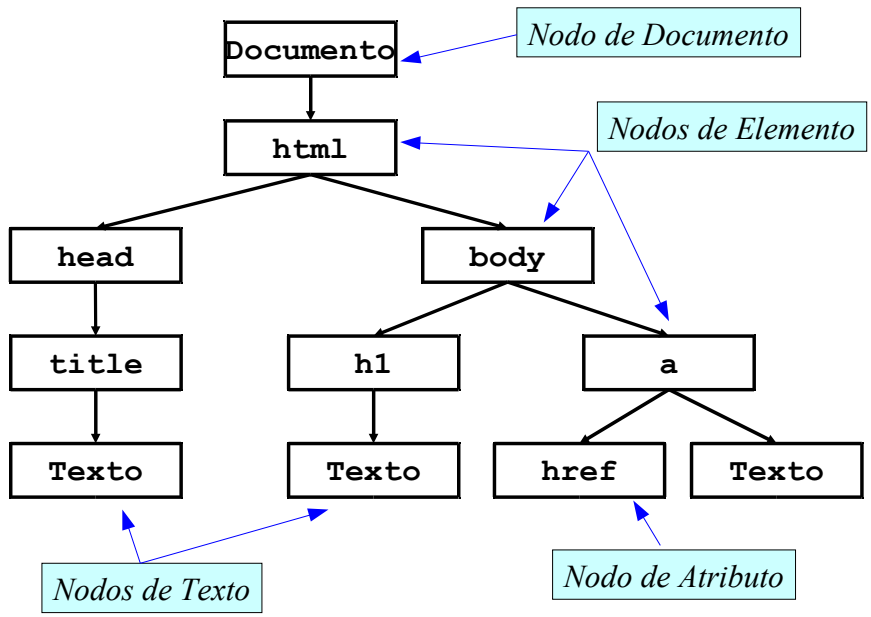


HTML DOM

- Exemplo:

```
<html>
  <head>
    <title>Teste DOM</title>
  </head>
  <body>
    <h1>Teste DOM 2</h1>
    <a href="teste.html">teste</a>
  </body>
</html>
```


HTML DOM



Propriedades e Métodos

- Nodos podem ser acessados via JavaScript
- Para obter um nodo (uma tag por exemplo):
  - **`document.getElementById(id)`** : retorna o elemento com o id passado

```
...
<p id="teste">Oi mundo</p>
...
x = document.getElementById("teste")
```


Propriedades e Métodos

- Nodos podem ser acessados via JavaScript
- Propriedades para um elemento `x`:
  - **`x.innerHTML`**: o texto dentro do elemento

```
<html><head><title>Teste</title></head>
<body>
<p id="teste">Oi mundo</p>
<script type="text/javascript">
   x = document.getElementById("teste")
   alert(x.innerHTML);              // mostra Oi Mundo
   x.innerHTML = "texto alterado"; // altera o parágrafo
</script>
</body></html>
```

Propriedades e Métodos

- Propriedades para um elemento `x`:
  - **`x.nodeName`**: é o nome de x
  - **`x.nodeValue`**: é o valor de x
  - **`x.parentNode`**: é o nodo pai de x
  - **`x.childNodes`**: é um array com todos os nodos filhos de x
  - **`x.firstChild`**: é o primeiro filho de x
  - **`x.lastChild`**: é o último filho de x
  - **`x.nextSibling`**: é o próximo irmão de x
  - **`x.attributes`**: é um array com todos os nodos atributos de x

Propriedades e Métodos

- Métodos para um elemento x:
  - **`x.getElementById(id)`**: retorna o elemento com o id passado
  - **`x.getElementsByTagName(name)`**: retorna todos os elementos de determinada tag
  - **`x.appendChild(node)`**: insere um nodo filho em x
  - **`x.insertBefore(node)`**: insere o elemento no pai, antes do nodo x
  - **`x.removeChild(node)`**: remove um nodo filho de x
  - **`x.replaceChild(newNode, oldNode)`**: substitui oldNode por newNode
  - **`x.setAttribute(name, value)`**: seta o atributo name com o valor value no nodo x

Exemplo

```
<html>
<head><title>Teste elementos</title></head>
<body>
<p id="prim">Texto Original do P</p>

<script type="text/javascript">
  x = document.getElementById("prim");
  document.write("Texto original: " + x.innerHTML);
  x.innerHTML = "***ALTERADO***";
  document.write("<br/>Texto alterado: " + x.innerHTML);
</script>

</body>
</html>
```


Exemplo

```
<html>
<head><title>Teste elementos</title></head>
<body>
<p id="prim">Texto Original do P</p>
<div id="meudiv"><h1 id="teste1">Teste 1</h1><h1 id="teste2">Teste
   2</h1><h1 id="teste3">Teste 3</h1></div>
<script type="text/javascript">
   x = document.getElementById("prim");
   document.write("Texto original: " + x.innerHTML);
   x.innerHTML = "***ALTERADO***";
   document.write("<br />Texto alterado: " + x.innerHTML);
   x = document.getElementById("teste1");
   document.write("<br />Conteudo H1: " + x.firstChild.nodeValue);
   x = document.getElementById("meudiv");
   document.write("<br />Primeiro: " + x.childNodes[0].innerHTML);
   document.write("<br />Primeiro: " + x.firstChild.innerHTML);
   document.write("<br />Ultimo: " +
        x.childNodes[x.childNodes.length -1].innerHTML);
   document.write("<br />Ultimo: " + x.lastChild.innerHTML);
</script>
</body>
</html>
```

Propriedades e Métodos

- Exemplo de appendChild, movendo elementos

```
<html><head><title>Teste</title></head>
<body>
<ul id="lista1"><li>Item 1</li><li>Item 2</li></ul>
<ul id="lista2"><li>Item 3</li><li>Item 4</li></ul>
<button onclick="mover()">Mover</button>
<script type="text/javascript">
   function mover() {
      x = document.getElementById("lista1").lastChild;
      document.getElementById("lista2").appendChild(x);
   }
</script>
</body>
</html>
```


Exercícios

- Executar os exemplos anteriores
- O exemplo anterior movia um item do <ul> de cima para baixo
- Adicionar um botão que move um item do <ul> de baixo para cima

Propriedades e Métodos

- Para criação de elementos:
  - **`x = document.createElement("p")`**: retorna um novo elemento p (parágrafo)
  - **`x = document.createTextNode("Texto")`** : retorna um novo elemento de texto
  - **`x = document.createAttribute("href")`** : retorna um novo nodo de atributo
    - Depois usar:

```
x.nodeValue = "http://www.google.com.br";
elem.setAttributeNode(x);
```


Propriedades e Métodos

- Exemplo de appendChild, criando elementos

```
<html><head><title>Teste</title></head>
<body>
<button onclick="criar()">criar</button>
<script type="text/javascript">
   function criar() {
      var p = document.createElement("p");
      var texto = document.createTextNode("Meu texto");
      p.appendChild(texto);
      document.body.appendChild(p);
   }
</script>
</body>
</html>
```

Propriedades

- Propriedades especiais:
  - **`document.documentElement`**: retorna o elemento raiz do documento
  - **`document.body`**: retorna diretamente o *body*

Propriedades de Nodos

- Propriedade **`nodeName`**
  - Só de leitura
  - Se o nodo for de elemento, é igual à tag (em maiúsculo)
  - Se o nodo for um atributo, é o nome do atributo
  - Se for um nodo de texto, é **`#text`**
  - Se for o nodo de documento, é **`#document`**

Propriedades de Nodos

- Propriedade **`nodeValue`**
  - Para nodos de elementos é indefinido
  - Para nodos de texto, é o próprio texto
  - Para atributos, é o valor do atributo
- Propriedade **`nodeType`**
  - Só de leitura
  - Retorna o tipo do nodo
    - Elemento: 1
    - Atributo: 2
    - Texto: 3
    - Comentário: 8
    - Documento: 9

Alteração de Propriedades

- Pode-se alterar a cor de fundo

```
document.body.bgColor = "yellow";
```

- Pode-se alterar o texto de um elemento

```
document.getElementById("texto").innerHTML="oi";
```

- Pode-se usar eventos para manipulação

```
<input type="button"
   onclick="document.body.bgColor='yellow';"
   value="Amarelo">
```

- Uso da propriedade style

```
document.body.style.backgroundColor = "blue";
```

Eventos..

- Ações detectadas pelo JavaScript
  - **`onLoad`**: quando o usuário entra na página
  - **`onUnload`**: quando o usuário sai da página
  - **`onFocus`**: quando o controle ganha o foco
  - **`onBlur`**: quando o controle perde o foco
  - **`onChange`**: quando o conteúdo do controle é alterado
  - **`onMouseOver`**: quando o mouse fica sobre o componente
  - **`onMouseOut`**: quando o mouse sai de cima do componente

CSS: Cascading Style Sheet

CSS

- *Cascading Style Sheet*
- Define como mostrar elementos HTML
- Podem ser definidos
  - Em cada elemento HTML
  - Em um estilo para a página
  - Em uma folha de estilos externa (.css)
- Múltiplas definições serão “cascateadas”

CSS

- Ordem de Aplicação
  1)Estilos *default* do navegador
  2)Folha de estilos externa (.css)
  3)Folha de estilos interna (da página)
  4)Estilo no elemento HTML (*inline*)
- Estilos escritos diretamente no elemento HTML (*inline*) têm prioridade sobre os demais

Sintaxe Básica

- Três partes
  - Seletor
  - Propriedade
  - Valor

```
seletor { propriedade: valor }
```

- Exemplo:
  - Incide sobre o elemento <body>

```
body { color: yellow }
```

  - Incide sobre o elemento <p>

```
p { text-align: center }
```


Declarar uma Folha de Estilos – EXTERNA (.css)

- Folha de estilos externa, na seção **<head>**:

```
<head>
  <link rel="stylesheet" type="text/css"
                href="estilo.css" />
</head>
```

- O arquivo .css pode ser escrito normalmente em um editor de textos. Por exemplo:

```
hr { color: sienna }
p { margin-left: 20px }
body { background-image: url("images/back40.gif") }
```

Declarar uma Folha de Estilos - INTERNA

- Folha de estilos interna, na seção <head>:

```
<head>
  <style type="text/css">
     hr {color: sienna }
     p {margin-left: 20px }
     body { background-image: url("images/back40.gif") }
  </style>
</head>
```


Declarar uma Folha de Estilos - INLINE

- Inline: Declaração dentro de um elemento HTML:

```
<p style="color: sienna; margin-left: 20px">Teste</p>
```

Múltiplas declarações

- Múltiplas definições para o mesmo elemento
- Exemplo: Folha externa

```
h3 {
   color: red;
   text-align: left;
   font-size: 8pt
}
```

- Exemplo: Folha interna

```
h3 {
   text-align: right;
   font-size: 20pt
}
```

- Um elemento **h3** com estas duas definições terá:

```
   color: red;
   text-align: right;
   font-size: 20pt
```


Sintaxe

- Se o valor tem várias palavras, coloca-se aspas:

```
p { font-family: "sans serif" }
```

- Se forem definidas várias propriedades, deve-se separar com ponto-e-vírgula:

```
p { text-align: center; color: red }
```

- Para melhorar a leitura, pula-se linhas:

```
p {
   text-align: center;
   color: red;
   font-family: arial
}
```

Sintaxe - Agrupamento

- Pode-se agrupar os elementos que devem ter o mesmo estilo

```
h1,h2,h3,h4,h5,h6 {
   color: green
}
```


Sintaxe - Classe

- Com o seletor de classe, pode-se definir estilos diferentes para o mesmo elemento

```
p.right { text-align: right }
p.left { text-align: left }
```

- Usa-se em HTML da seguinte forma:

```
<p class="right">Meu texto alinhado</p>
<p class="left">Meu texto alinhado</p>
```

Sintaxe - Classe

- Para aplicar mais de uma classe a um elemento, usa-se:

```
p.right { text-align: right }
p.bold { font-weight: bold }

<p class="right bold">Meu texto alinhado e negrito</
  p>
```


Sintaxe - Classe

- Para aplicar uma classe a qualquer elemento, usa-se:

```
.center { text-align: center }

<p class="center">Meu texto centralizado</p>
<h1 class="center">Outro texto centralizado</h1>
```

Sintaxe – Seletor por Propriedade

dhtml#23

- Para aplicar uma classe a um elemento com uma determinada propriedade, usa-se:

```
input[type="text"] { background-color: blue }

<input type="text" value="Teste" />
```


Sintaxe – Seletor por Id

dhtml#24

- Pode-se selecionar por Id do elemento, usando “#”:

```
#green { color: green }

<input id="green" type="text" value="Teste" />
```

- Para selecionar por elemento e Id, usa-se

```
p#para1 {
 text-align: center;
 color: red
}

<p id="para1">Teste</p>
```

Sintaxe - Comentários

- Usa-se /* e */

```
/* Primeiro Comentário */
p {
  text-align: center;
  /* Outro Comentário */
  color: black;
  font-family: arial
}
```


Propriedades de *Background*

- Seta o fundo dos elementos
  - **`background-color`**: seta a cor de fundo
  - **`background-attachment`**: se a imagem de fundo é fixa ou rola com o resto da página
  - **`background-image`**: seta a imagem de fundo
    - **`url(...)`**
  - **`background-position`**: posição inicial da imagem de fundo
    - **`top left, top center, ..., x y`**
  - **`background-repeat`**: seta se a imagem será repetida
    - **`repeat, repeat-x, repeat-y, no-repeat`**

Propriedades de Texto

- Seta características de texto
  - **`color`**: cor do texto
  - **`direction`**: direção
    - **`ltr, rtl`**
  - **`line-height`**: distância entre linhas
  - **`letter-spacing`**: espaço entre caracteres
  - **`text-align`**: alinha o texto
    - **`left, right, center, justify`**
  - **`text-decoration`**: decoração do texto
    - **`none, underline, overline, line-through, blink`**
  - **`text-indent`**: identa a primeira linha do texto
  - **`text-transform`**: controla as letras
    - **`none, capitalize, uppercase, lowercase`**
  - **`white-space`**: tratamento de espaço em branco
  - **`word-spacing`**: seta espaço entre palavras

Propriedades de Fonte

- Usadas para definir características da fonte
  - **`font-family`**: lista de fontes do texto
  - **`font-size`**: tamanho da fonte
    - **`xx-small, x-small, small, medium, large, ..., <tam>`**
  - **`font-style`**: estilo da fonte
    - **`normal, italic, oblique`**
  - **`font-weight`**:
    - **`normal, bold, bolder, lighter, 100~900`**

Propriedades de Bordas

- Definição de bordas dos elementos
  - **`border-color`**: cor da borda
  - **`border-style`**: estilo da borda
    - **`none, hidden, dotted, dashed, solid, …`**
  - **`border-width`**: tamanho da borda
    - **`thin, medium, thick, <tam>`**
  - Para **`XXX`** sendo **`left, right, bottom, top`**:
    - **`border-XXX-color`**
    - **`border-XXX-style`**
    - **`border-XXX-width`**

Propriedades de *Outline*

- Para destacar um elemento
- Linha FORA da borda
  - **`outline-color`**: cor do destaque
    - **`<cor>, invert`**
  - **`outline-style`**: estilo
    - **`none, dotted, dashed, solid, double, groove, ridge, inset, outset`**
  - **`outline-width`**: tamanho do destaque
    - **`thin, medium, thick, <tamanho>`**

Propriedades de Margem

- Seta a margem dos elementos
  - **margin-bottom**
    - **auto, <tamanho>, %**
  - **margin-left**
    - **auto, <tamanho>, %**
  - **margin-right**
    - **auto, <tamanho>, %**
  - **margin-top**
    - **auto, <tamanho>, %**

Propriedades de *Padding*

- Distância entre a borda e o conteúdo
  - **padding-bottom**
    - **<tamanho>, %**
  - **padding-left**
    - **<tamanho>, %**
  - **padding-right**
    - **<tamanho>, %**
  - **padding-top**
    - **<tamanho>, %**

Propriedades de Listas

- Define listas e estilos de listas
  - **`list-style-image`**: imagem como marcador
    - **`none, url`**
  - **`list-style-position`**: onde o marcardor é colocado
    - **`inside, outside`**
  - **`list-style-type`**: seta o tipo do marcador
    - **`none, disc, circle, square, decimal, ...`**

Propriedades de Tabelas

- Seta características de tabelas
  - **`border-collapse`**: se a borda será padrão HTML ou mostrada como uma linha só
    - **`collapse, separate`**
  - **`border-spacing`**: distância entre as bordas separadas (separate)
    - <tamanho>
  - **`caption-side`**: posição da legenda
    - **`top, botton, left, right`**
  - **`empty-cells`**: se serão mostradas células em branco
    - **`show, hide`**