Introdução ao JavaScript

Tiago Lopes Telecken
telecken@gmail.com

Tópicos

- Características
- Comandos
- Variáveis
- Operadores matemáticos
- Operadores lógicos
- Estruturas condicionais
- Estruturas de repetição
- Funções
- Objetos
- Eventos

Introdução - JavaScript

- **Java Script é uma linguagem para criação de scripts em clientes Web;**
  - **Criação de interações entre usuários e páginas HTML**
  - **Acesso e manipulação de recursos de navegadores Web e páginas HTML**
  - **Criação de animações e recursos visuais**
  - **Validação e teste de formulários**
  - **Realização de processamentos e cálculos diversos em navegadores Web**

Características

- Linguagem compacta baseada em objetos, específica para aplicações web.
- Escrita diretamente no HTML através de qualquer editor de textos simples, como por exemplo, Bloco de Notas.
- Reconhece situações (eventos) como cliques, movimentos do mouse, entrada de dados, etc.
- Sintaxe parecida com a linguagem Java, porém é mais simples e flexível. Por outro lado tem menos recursos

Características

- **Linguagem Segura!**
  - **Não possui autorização para gravar dados no disco rígido.**
- **Linguagem interpretada**
  - **Não há a necessidade de compilar o código**
  - **É só escrever o código e executar**
  - **Agiliza o desenvolvimento e a depuração**
  - **Desempenho mais lento**
- **Possue objetos e funções**

Características

- **Há um esforço para que o JS funcione de forma padronizada em todos os navegadores. Na maioria das situações há esta compatibilidade**
- **Entretanto ainda há diferenças de funcionamento/compatibilidade**
- **Nos piores casos os navegadores nem rodam JS (como nas situações abaixo)**
  - Usar um navegador antigo ou raro com suporte DOM incompleto ou incomum.
  - Usar um navegador de um PDA ou telefone móvel que não está apto a executar JavaScript.
  - Ter a execução do JavaScript desabilitada por normas de segurança.
  -

Comandos

- **É case-sensitive (diferencia maiúsculas de minúsculas).**
- **Comandos terminam com “;”**
- **Comentários //   /* */**

Primeiros programas

- Inserindo um código JavaScript em uma página html. Existem 3 maneiras
  - **1 Utilizar a tag <script>**

```
<html>
   <head></head>
   <body>
     <script language="JavaScript">
              document.write(“Ola mundo!”);
     </script>
   </body>
</html>
```

Primeiros programas

```
<html>     <head></head>
   <body>
     Um <br>
     <script language="JavaScript">
              document.write(“Ola mundo!”);
     </script>
<br>dois
     <script language="JavaScript">
              document.write(“<br><B>Ola mundo!</B>”);
     </script>
   </body>
</html>
```

Comando document.write

- document.write(“”)
- Insere um código na página HTML, pode ser uma string simples ou pode conter tags

```
document.write(“Ola mundo!”);
document.write(“<I><B>Ola mundo!</B></I>”);
```

Primeiros programas

- Usando arquivos externos
  - **2 Utilizar o atributo “src” da tag <script>**

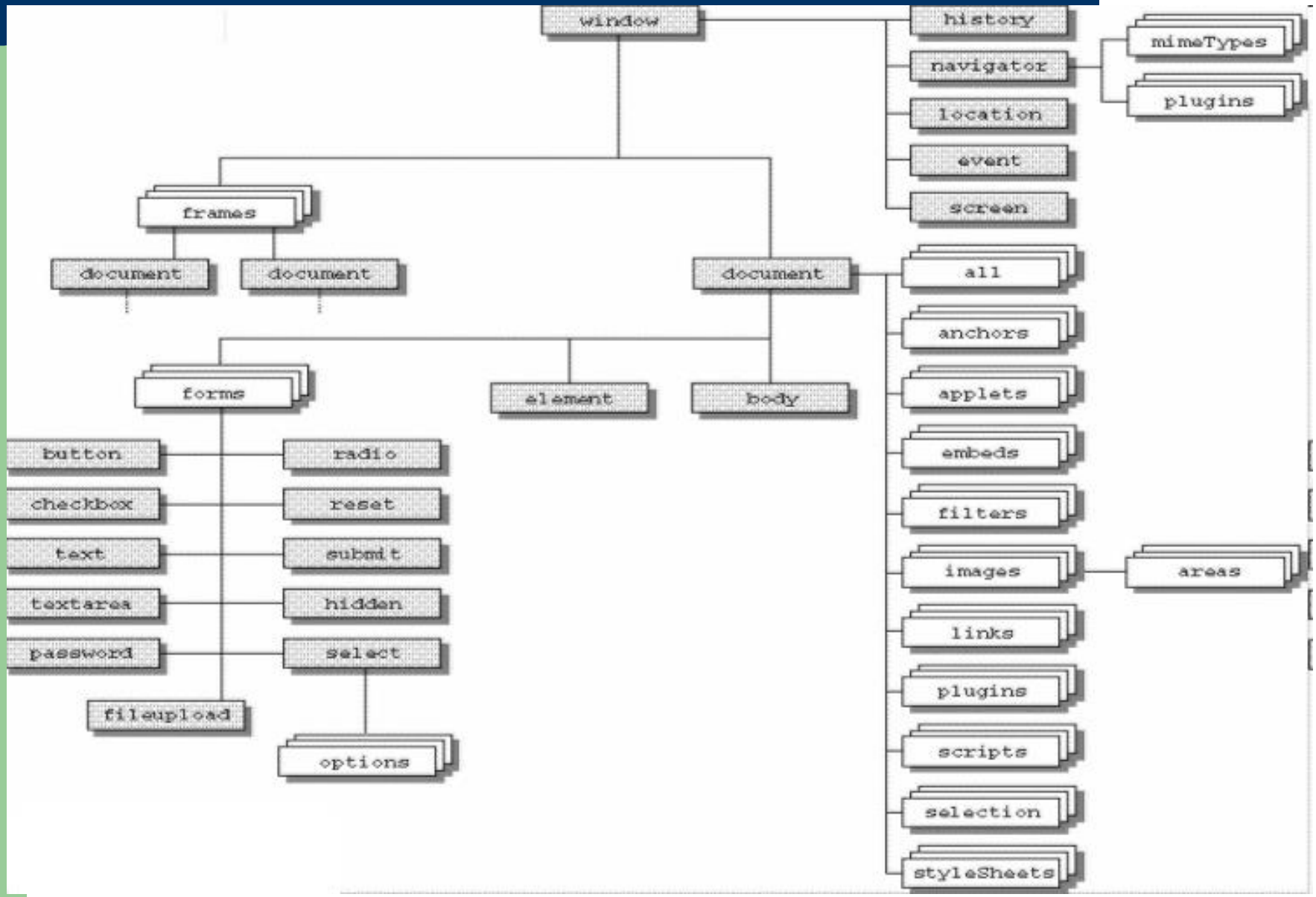
```
<html>
   <head></head>
   <body>
     <script language="JavaScript" src“../file/ola.js”>
     </script>
   </body>
</html>
```

  - **3 Utilizar atributos de alguns elementos do HTML**
    - Será visto quando trabalharmos com eventos

Objetos JavaScript

- Objetos, propriedades, métodos
  - carro
  - carro.ano
  - carro.dono.nome
  - document.write(“ola”) // window.document.write(“ola”)
  - navigator.appName; //window.navigator.appName;
- Objetos pré-existentes, criados pelo usuário

Hierarquia de objetos



Propriedades de Objetos JavaScript

```
<html>
    <head>
        <title>Untitled</title>
    </head>
    <body>
        <Script Language=JavaScript>
          document.write (“Seu browser é “ + navigator.appName);
        </Script>
    </body>
</html>
```

Propriedades de Objetos JavaScript

```
<html>
    <head>
        <title>Untitled</title>
    </head>
    <body>
        <Script Language="JavaScript">
          document.write ("<br>Navegador: " + navigator.appName);
         document.write ("<br> Versão do Navegador: "+ navigator.appVersion);
          document.write ("<br> Sistema operacional : "+ navigator.platform);
           document.write ("<br> location.href : "+ location.href);
          document.write ("<br>Tela Largura: "+ screen.width);
          document.write ("<br> Tela altura : "+ screen.height);
         </Script>
    </body>
</html>
```

Métodos de Objetos JavaScript

- Além de uma página

```
<html>
<head>
    <script>
            window.alert(“Mensagem para o  usuário");
            window.confirm(“Aceita as condições?”);
            window.prompt(“Qual o seu nome?”,”fulano”);
            window.open(“http://www.google.com”,”janela2”);
    </script>
</head>
<body>   </body>
</html>
```

Métodos de Objetos JavaScript

```
<html>
<head>
    </head>
<body>
    <script>
        nome= prompt(“Qual o seu nome?”,”fulano”);
        document.write(“Seja bem vindo ”+nome);
    </script>
</body>
</html>
```

Funções matemáticas

- **Math.abs(número)** - retorna o valor absoluto do número (ponto flutuante)
- **Math.ceil(número)** - retorna o próximo valor inteiro maior que o número
- **Math.floor(número)** - retorna o próximo valor inteiro menor que o número
- **Math.round(número)** - retorna o valor inteiro, arredondado, do número
- **Math.pow(base, expoente)** - retorna o cálculo do exponencial
- **Math.max(número1, número2)** - retorna o maior número dos dois fornecidos
- **Math.min(número1, número2)** - retorna o menor número dos dois fornecidos
- **Math.sqrt(número)** - retorna a raiz quadrada do número

- alert (Math.sqrt(16))

Manipulando Strings

- As variáveis de tipo texto são objetos da classe String. Possuem propriedades e métodos.
- **Propriedade Length:** armazena o número de caracteres do String.
- **Métodos**
- **charAt(índice)**: Devolve o caractere que há na posição indicada como índice. As posições de um string começam em 0.
- **substring(início,fim):** Devolve o substring que começa no caractere de início e termina no caractere de fim.
- **toLowerCase():** Coloca todos os caracteres de um string em minúsculas.
- **toUpperCase():** Coloca todos os caracteres de um string em maiúsculas.
- **toString():** converte numeros e outros tipos de dados em strings.

Manipulando Strings

- <html>
- <body>

- <script type="text/javascript">
- var txt="Hello World!";
- document.write(txt.length);
- </script>

- </body>
- </html>

- **txt.charAt(3) // l**
- **txt.substring(2,5) // llo**

Tipos de dados

- String
  - A=“ola”;
  - B=‘ ola2 \n \t ’;
- Booleano
  - true //ou qualquer string diferente de 0 ou string vazia
  - false //ou 0 ou string vazia
- Número
  - Inteiro, ponto flutuante, decimal, octal, hexadecimal
  - 1; 1.2, 1.54e5, 0045, 0x66f
  - NaN, infinito
- undefined- propriedade que nao existe,variavel sem valor atribuido
- null – variável sem valor

Variáveis

- As variáveis são criadas automaticamente quando valores são atribuidos a estas variáveis
  - Não há a necessidade de se declarar as variáveis antes delas serem usadas
  - Os tipos das variáveis são deduzidos pelo interpretador JavaScript
- Exemplos

Valor = 30;	int
Nome=“Fulano”;	string
Peso= 20.5	float

- Opcionalmente pode-se declarar a existência de uma variável antes de usa-la
  - var Peso;

Variáveis

- Os tipos das variáveis podem ser alterados durante a execução de um código JavaScript

```
Carga = 30;                 // tipo int , valor 30
Peso= 20.5;                 // tipo float, valor 20.5
Carga= Carga + Peso;        // tipo float, valor 50.5
Carga=Carga + ”oi”;         // tipo string, valor “50.5oi”
```

- Existem funções que convertem tipos
  - parseInt(“10”);
  - parseFloat(“44.6”);

Operadores Aritméticos

+ adição de valor e concatenação de Strings;
- subtração de valores;
* multiplicação de valores;
/ divisão de valores;
=, += atribuição

- Utilizados em cálculos e manuseio de variáveis.
  - Peso= Peso+20*4
  - Nome=“fulano”+”de Tal”

Operadores lógicos

- São operadores a serem utilizados em estruturas condicionais e de repetição

==	Igual
!=	Diferente
>	Maior
>=	Maior ou Igual
<	Menor
<=	Menor ou Igual
&&	E
‖	Ou

- Peso == Carga //retorna verdadeiro ou falso

Estruturas Condicionais

- São comandos que condicionam a execução de certa tarefa à veracidade ou não de uma determinada condição
- If, Else

```
if (condição) {
     ação para condição satisfeita
}
else {
    ação para condição não satisfeita
 }
```

```
if (Idade < 18) {
    Categoria = "Menor“;
}
else {
    Categoria = "Maior“;
}
```

Estruturas de repetição

- Estruturas que permitem que uma parte do código seja executada várias vezes
- While
  - Executa uma ação enquanto determinada condição for verdadeira.

```
while (condição)
 {
    ação
}
```

```
x=0;
while (x<3)
 {
     alert ("X igual a " + x);
     x=x+1;
}
```

Estruturas de repetição

- For

```
for ( [inicialização de variável de controle ;] [condição ;]
    [incremento da variável de controle] ) {
        comandos
}
```

```
for (x = 0 ; x < 3 ; x++)
{
     alert ("X igual a " + x);
}
```

Funções do programador

- Funções modularizam os programas
- Programa modularizado em funções torna-se mais fácil de manter;

```
function nomeDaFunção ( parametro1,... parametroN ){
        instruções;
        return valor;
}

function soma (a, b){
       return a+b;
}
```

Funções

```
<html>
<head>
  <script>
     function soma (a, b) {
        return a+b;
     }
  </script>
</head>
<body>
  <script>
     alert (soma(1, 2) );
  </script>
</body>
</html>
```

Escopo

- Variavel declarada dentro de uma função só é vista dentro da função
- Variável declarada fora de uma função pode ser acessada por comandos que estão dentro de qq função e fora de funções (visibilidade global)
- Funções tem visibilidade global
- As variáveis/funções não são visíveis de uma página para outra

HTML - DOM

- Document Object Model
- API de acesso e manipulação de documentos HTML

DOM

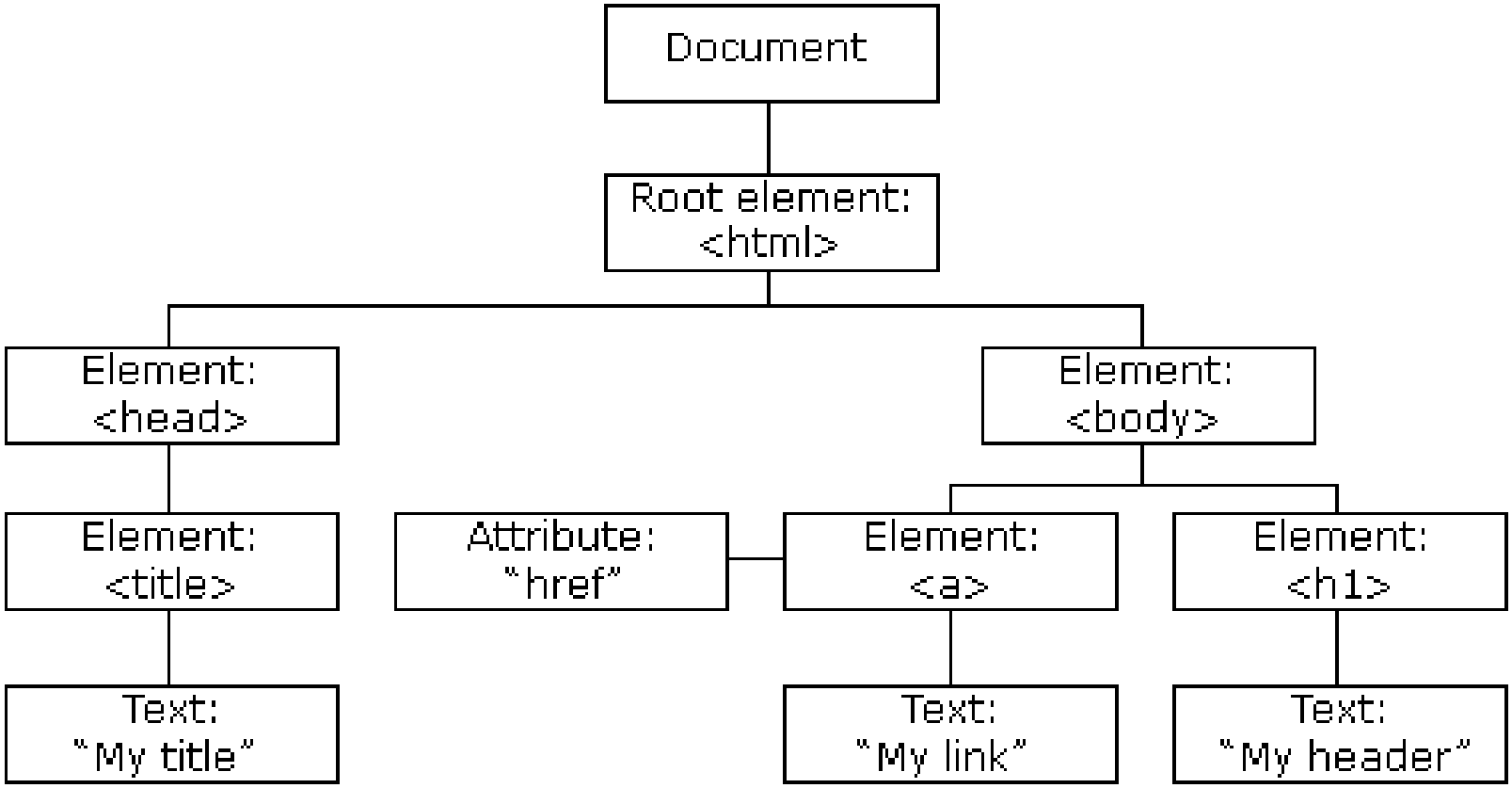
- O DOM transforma todo documento HTML em objetos com propriedades e métodos que podem ser acessados
- Para o DOM, tudo em um documento HTML é um nodo (o nodo é um objeto com propriedades e métodos).
  - O documento inteiro é um nodo
  - Todos elementos HTML são nodos
  - Todos textos nos elementos HTML são nodos texto
  - Todos atributos são nodos
  - Todos comentários são nodos

DOM – Nodos (Node)

- **Exemplo**
- <html>
  <head>
  <title>DOM Tutorial</title>
  </head>
  <body>
  <h1>DOM 1</h1>
  Ola Mundo!
  </body>
  </html>

- </html> é o nodo raiz (root)
- Todos os outros nodos estão dentro do nodo HTML

A árvore DOM de nodos



Acessando nodos - getElementById()

- **O método getElementById()**
  - Retorna o elemento com a ID especificada:
  - *node*.getElementById(*“id"*)
  - document.getElementById(“menu");
  - Retorna o elemento com id=“menu":

```
<html>
<body>
    <p id="intro">Hello World!</p>
    <script type="text/javascript">

        x=document.getElementById("intro");
        document.write(x.firstChild.nodeValue);

    </script>
</body>
</html>
```

Neste exemplo foi utilizado o **firstChild.nodeValue**

Acessando nodos - getElementById()

- Neste exemplo o nodo com id=“meuTitulo” será capturado pelo comando document.getElementById("meuTitulo") e seu conteudo sera jogado na variavel x. Depois o comando alert(x.innerHTML) vai mostrar uma mensagem com o conteudo de x. innerHTML retorna o conteúdo de um nodo, ou seja o texto que esta dentro da tag.

```
<html>
<head>
</head>
<body>
<h1 id="meuTitulo">Este título será capturado pelo getElementById e
    mostrado na mensagem alert</h1>
<h1 id="OutroTitulo">Este título não sera capturado</h1>
      <script type="text/javascript">
            var x=document.getElementById("meuTitulo");
            alert(x.innerHTML);
      </script>
</body>
</html>
```

- Para elementos, innerHTML é equivalente a firstChild.nodeValue

Acessando nodos - getElementsByTagName()

- **O método getElementsByTagName()**
  - Retorna todos os elementos com o nome de tag especificado
  - *node*.getElementsByTagName(*"tagname"*);
  - document.getElementsByTagName("p");
  - Retorna uma lista de todos elementos <p> do documento
- O código a seguir retorna uma lista de nodos com todos elementos <p> que são descendentes do elemento com id="main"
  - document.getElementById('main').getElementsByTagName("p");

Acessando nodos - getElementsByTagName()

- **O método getElementsByTagName() retorna uma lista de nodos. Esta lista é um array de nodos**
- **Selecionar todos nodos <p>**
  - x=document.getElementsByTagName("p");
- Acessar o segundo p da lista
  - y=x[1];
- Tamanho da lista
  - X.length

Acessando nodos - getElementsByTagName()

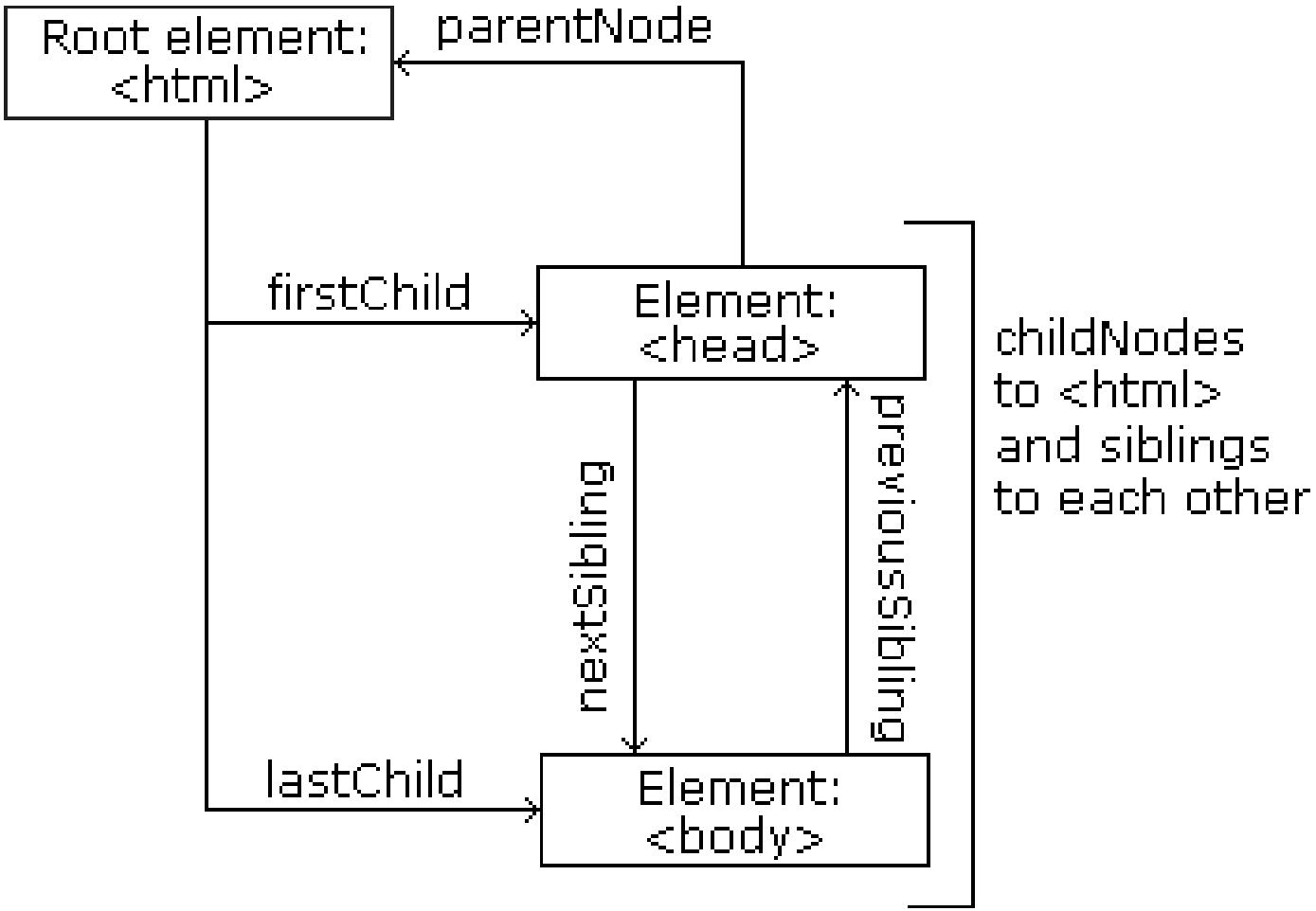
- Na primeira linha do script, x recebe uma lista com todos os elementos de uma pg HTML.
- Depois é definido um laço que vai de 0 ao número de “p”s que estão na lista de x
- Dentro do laço, um comando escreve o conteúdo de cada p. O comando innerHTML retorna o que esta escrito dentro de um elemento. Neste caso o que está escrito entre e

```
<html><body>
<p>primeiro</p>
<p>segundo</p><hr>
<script>
    x=document.getElementsByTagName("p");
    for (i=0;i<x.length;i++)
    {
    document.write(x[i].innerHTML);
    document.write("<br />");
    }
</script>
</body></html>
```

Acessando nodos – Por relacionamentos

- Os nodos possuem relacionamentos entre si, conforme sua posição na árvore
- Um nodo pode ter:
  - Filhos (child)
  - Pai (parent)
  - Irmaos (siblings)

Acessando nodos – Por relacionamentos



Acessando nodos – Por relacionamentos

- <html>
  <head>
  <title>DOM Tutorial</title>
  </head>
  <body>
  <h1>DOM 1</h1>
  Ola Mundo!
  </body>
  </html>

Acessando nodos – Por relacionamentos

- Navegando pelos relacionamentos
- Propriedades parentNode, firstChild e lastChild, nextSibling e previousSibling
- <html>
  <body>
  Hello World!
  <script type="text/javascript">
  x=document.getElementById("intro");
  document.write(x.firstChild.nodeValue);
  </script>
  </body>
  </html>

Acessando nodos – Por atalhos

- document.documentElement – retorna o elemento raiz <html>
- document.body – retorna o elemento <body>
- Pelo atributo **name**, pela palavra reservada **this**, entre outros… (exemplos adiante em eventos)

Acessando Propriedades

- **nodeName**
- Especifica o nome dos nodos (depende do tipo de nodo).
  - Para um elemento node é o nome da tag
  - Para um atributo é o nome do atributo
  - Para texto é sempre #text
- **nodeValue**
- Especifica o valor de um nodo.
  - Indefinido para elementos
  - O próprio texto para textos
  - O valor dos atributos para atributos
- <html>
  <body>
  Hello World!
  <script type="text/javascript">
  x=document.getElementById("intro");
  document.write(x.firstChild.nodeValue);
  </script>
  <html>
  <body>

Acessando Propriedades

- **nodeType**
- Retorna o tipo do nodo

Tipo de nodo	**NodeType**
Element	1
Attribute	2
Text	3
Comment	8
Document	9

Acessando Propriedades

```
<html>
<head></head>
<body>
<p id="intro">Alo Mundo!</p>
<hr><hr><hr><br>
        <script type="text/javascript">
                        x=document.getElementById("intro");
                        //propridades do elemento body
                        document.write(x.parentNode.nodeName +"<br>");
                        document.write(x.parentNode.nodeValue +"<br>");
                        document.write(x.parentNode.nodeType +"<br>");
                        document.write("<HR>");
                        // propridades do elemento P
                        document.write(x.nodeName +"<br>");
                        document.write(x.nodeValue +"<br>");
                        document.write(x.nodeType +"<br>");
                        document.write("<HR>");
                        // propridades do texto Alo mundo
                        document.write(x.firstChild.nodeName +"<br>");
                        document.write(x.firstChild.nodeValue +"<br>");
                        document.write(x.firstChild.nodeType +"<br>");
        </script>
</body></html>
```

Alterando documentos

- **Alterando atributos**
- <html>
  <body>
  <script type="text/javascript">
  document.body.bgColor="yellow";
  </script>
  </body>
  </html>
- **Usando o innerHTML**
- <html>
  <body>
  Hello World!
  <script type="text/javascript">
  document.getElementById("p1").innerHTML="New text!";
  </script>
  </body>
  </html>

Alterando documentos

- **Usando eventos**
- <html>
  <body>
  <input type="button"
  onclick="document.body.bgColor='lavender';"
  value="Change background color" />
  </body>
  </html>

Alterando documentos

- **Usando o objeto style //altera prop. CSS**
- <html>
  <head>
  <script type="text/javascript">
  function ChangeBackground()
  {
  document.body.style.backgroundColor="lavender";
  }
  </script>
  </head>

  <body>
  <input type="button" onclick="ChangeBackground()"
  value="Change background color" />
  </body>
  </html>

Alterando documentos

```
<html>
<head>
<script type="text/javascript">
function ChangeStyle()
{
document.getElementById("p1").style.color="blue";
document.getElementById("p1").style.fontFamily="Arial";
}
</script>
</head>

<body>
<p id="p1">Hello world!</p>
<input type="button" onclick="ChangeStyle()" value="Change
style" />
</body>
</html>
```

Eventos e manipuladores de eventos

- Eventos: São fatos que ocorrem durante a interação do usuário com a página (clicar botões, selecionar caixas de texto, carregar páginas, etc).
- Manipuladores de eventos: são atributos de elementos HTML que detectam quando determinados eventos ocorrem e podem disparar ações quando estes eventos ocorrem.
- A seguir uma lista de manipuladores de eventos indicando os eventos que podem detectar. Também é mostrado os elementos HTML onde estes eventos podem ocorrer e consequentemente onde os manipuladores podem ser colocados.

- **onload** - Detecta a carga do documento. Ou seja, quando o usuário acessa a página.
  - Válido para o elemento Body
- **onunload** - Detecta quando o usuário sai da página.
  - Válido para o elemento Body
- **onchange** - Detecta quando o objeto perde o foco e houve mudança de conteúdo (o usuário selecionou um item ou escreveu um novo texto em uma caixa de texto).
  - válido para o Text, Select e Textarea.

Eventos

- **onblur** - Detecta quando o objeto perde o foco, independente de ter havido mudança.Por exemplo quando o usuário clica em outra página ou elemento de formulário
  - válido para o Text, Select e Textarea.
- **onfocus** - Detecta quando o objeto recebe o foco. Ou seja quando o usuário clica no objeto ou o seleciona através do teclado.
  - válido para o Text, Select e Textarea.
- **onclick** - Detecta quando o objeto recebe um Click do Mouse.
  - válido para o Button, Checkbox, Radio, Link, Reset e Submit.
- **onmouseover** - Detecta quando o ponteiro do mouse passa por sobre o objeto.
  - válido para Links.
- **onselect** - Detecta quando o objeto é selecionado.
  - válido para o Text e Textarea.
- **onsubmit** - Detecta quando um botão tipo Submit recebe um click do mouse.
  - válido para o Form.

Eventos

- Os manipuladores de eventos citados anteriormente são atributos que podem ser colocados em determinados elementos HTML. Eles vão cuidar se determinado evento ocorre no elemento onde foram inseridos
- No exemplo abaixo o manipulador/atributo onchange vai detectar se o usuário muda o texto que esta escrito na caixa de texto “CxTexto”. Se isto ocorrer ele dispara uma ação (mostra uma mensagem de alerta).
- O valor do atributo é um comando javascript que será disparado quando ocorrer o evento monitorado pelo manipulador

- <form name="Text">
  Entrada de Texto
  <input type=“text” size=“20” name="CxTexto“ value=""
  onchange="alert ('Voce digitou ' + CxTexto.value)">
</form>

Eventos

- Ao ocorrer um evento o comando JavaScript determina a mudança de uma propriedade do objeto document

<input type=“radio” name="Rad" value="1“ onclick="document.bgColor='green'"> Fundo Verde
<input type=“radio” name="Rad" value="2"
onclick="document.bgColor='blueviolet'"> Fundo Violeta
<input type=“radio” name="Rad" value="3"
onclick="document.bgColor='#FFFF00'"> Fundo Amarelo

Eventos

```
<script>
function TestaVal() {
      if (document.TesteSub.Teste.value == "") {
            alert ("Campo nao Preenchido...Form nao Submetido");
      return false; }
      else {
      alert ("Tudo Ok....Form Submetido");
      return true; } }
</script>

<form method="POST" name="TesteSub“ onSubmit="return TestaVal()"
action="localhost/local.php">
Digite um Texto <input type=text size=10 maxlength=10 name="Teste“ value="">
Botao Submit <input type=submit name="Bsub" value="Manda p/Server">
</form>
```

Eventos

- O comando **this** referencia o objeto onde o **this** está inserido. No exemplo abaixo o "Combo2"

```
<script>
    function Vermult(Lista) {
            var opcoes = ""
            for (i = 0 ; i < Lista.length ; i++) {
                    if (Lista.options[i].selected) {
                            opcoes += (Lista.options[i].value + ", ")
                     }
            }
            alert ("As opcoes escolhidas foram : " + opcoes)
    }
</script>
Objeto Select2
<select name="Combo2" size=4 multiple onblur="Vermult(this)">
    <option value="List1">Escolha 1 </option>
    <option value="List2">Escolha 2 </option>
    <option value="List3">Escolha 3 </option>
    <option value="List4">Escolha 4 </option>
</select>
```

Eventos

- Cuidar o modo como os elementos HTML são referenciados

```
<SCRIPT>
function AltMaiusc () {
document.TCheck.Muda.value = document.TCheck.Muda.value.toUpperCase()
document.TCheck.Opt1.checked = false
}
function AltMinusc () {
document.TCheck.Muda.value = document.TCheck.Muda.value.toLowerCase()
document.TCheck.Opt2.checked = false
}
</SCRIPT>
<form name="TCheck">
    Muda Case <input type=text size=20 maxlength=20 name="Muda">
    Minusculo <input type=checkbox name="Opt1" value="1" checked
    onclick="if (this.checked) { AltMinusc() } ">
    Maiusculo <input type=checkbox name="Opt2" value="2“ onclick="if
    (this.checked) { AltMaiusc() } ">
    Demo valor <input type=checkbox name="Opt3“ onclick="if
    (Opt3.checked){alert ('Server recebera = ' + Opt3.value) } ">
</form>
```

Eventos

- onmouseover: quando o mouse passa por cima
- onmouseout: quando o mouse sai de cima
- <html>
  <head>
  <script type="text/javascript">
  function mouseOver()
  {
  document.getElementById("b1").src ="b_blue.gif";
  }
  function mouseOut()
  {
  document.getElementById("b1").src ="b_pink.gif";
  }
  </script>
  </head>

  <body>

  <img border="0" alt="Visite o Google!" src="b_pink.gif" id="b1"
  onmouseover="mouseOver()" onmouseout="mouseOut()" />
  </body>
  </html>

Provocando, simulando eventos

- Além de detectar eventos o javascript pode simula-los. Isto é útil para forçar que algum evento ocorra sem depender do usuário final
- A seguir a lista de métodos que simulam eventos
  - blur(): tira o foco do elemento
  - click(): simula um click do botao
  - focus(): coloca o foco em um elemento
  - reset (): reseta um formulário
  - submit (): submete um formulário sem que o usuário clique o botão submit
  - select (): seleciona um elemento

Simulando eventos

```
<html>
<head>
<script type="text/javascript">
function setFocus()
{
document.getElementById("fname").focus();
}
</script>
</head>
<body onload="setFocus()">
<form>
Name: <input type="text" id="fname" size="30"><br />
Age: <input type="text" id="age" size="30">
</form>
</body>
</html>
```

- Ao carregar a página o foco é colocado na primeira caixa de texto. Por isso ao carregar a página o cursor já fica nesta caixa de texto. Foi usado o método focus().

Eventos com tempo

- setTimeout (“ação”,milesegundos);
- Dispara uma ação após o tempo informado
- 1 segundo tem 1000 milesegundos

- ```
  <html>
  <head>
  <script type="text/javascript">
  function timedMsg()
  {
  var t=setTimeout("alert('5 segundos depois!')",5000);
  }
  </script>
  </head>

  <body>
  <form>
  <input type="button" value="Display timed alertbox!"
  onClick="timedMsg()" />
  </form>
  </body>
  </html>
  ```
  ```

Eventos com tempo

- setTimeout (“ação”,milesegundos);
- Dispara uma ação quando passa o tempo informado (1 vez)
- clearInterval é usado para interromper a chamada de funções (setInterval)

```
<html>
<head>
<script type="text/javascript">
function timedMsg()
{
 t=setInterval("alert(‘2 segundos depois!')",2000);
}
function parar(){
clearInterval(t);
}
</script>
</head>

<body>
<form>
<input type="button" value="Display timed alertbox!"
onClick="timedMsg()" />
<input type="button" value=“Parar"
onClick=“parar()" />
</form>
</body>
</html>
```

Eventos com addEventListener

- document.getElementById(“teste").addEventListener(“click", funcao, true);
- Num mesmo elemento podem ser adicionados vários event handlers
- Click: evento que dispara uma funcao
- Funcao: funçao disparada
- True: é o padrão e define que os elementos internos disparam eventos primeiro. False determina que os eventos dos elementos externos são executados primeiro

```
<div><p>oi</p></div>
```

- document.getElementById(“teste").removeEventListener(“click", funcao);
  - Remove o event handler

Mais Eventos

- document.getElementById(“teste").onclick = funcao;
  - Forma alternativa
- window.addEventListener("keyup", funcao, false);
  - Window é o objeto pai do JS. Neste caso se a qq momento uma tecla é apertada a funcao é disparada

Mais Eventos

```
window.addEventListener("keydown", verifica, false);


function verifica(e) {
    if (e.keyCode == "65") {
        alert(“A tecla 'a' foi pressionada.");
    }
}
```

Quando um event handler chama uma função ele passa para a funçao o objeto “e” que é o evento. Cada evento tem suas propriedades. No exemplo acima o evento clicar tecla tem a propriedade que retorna o código da tecla clicada

Introdução ao JavaScript

Tiago Lopes Telecken
telecken@gmail.com