

Introdução ao JavaScript

Tiago Lopes Telecken

tiago.telecken@riogrande.ifrs.edu.br



HTML - DOM

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

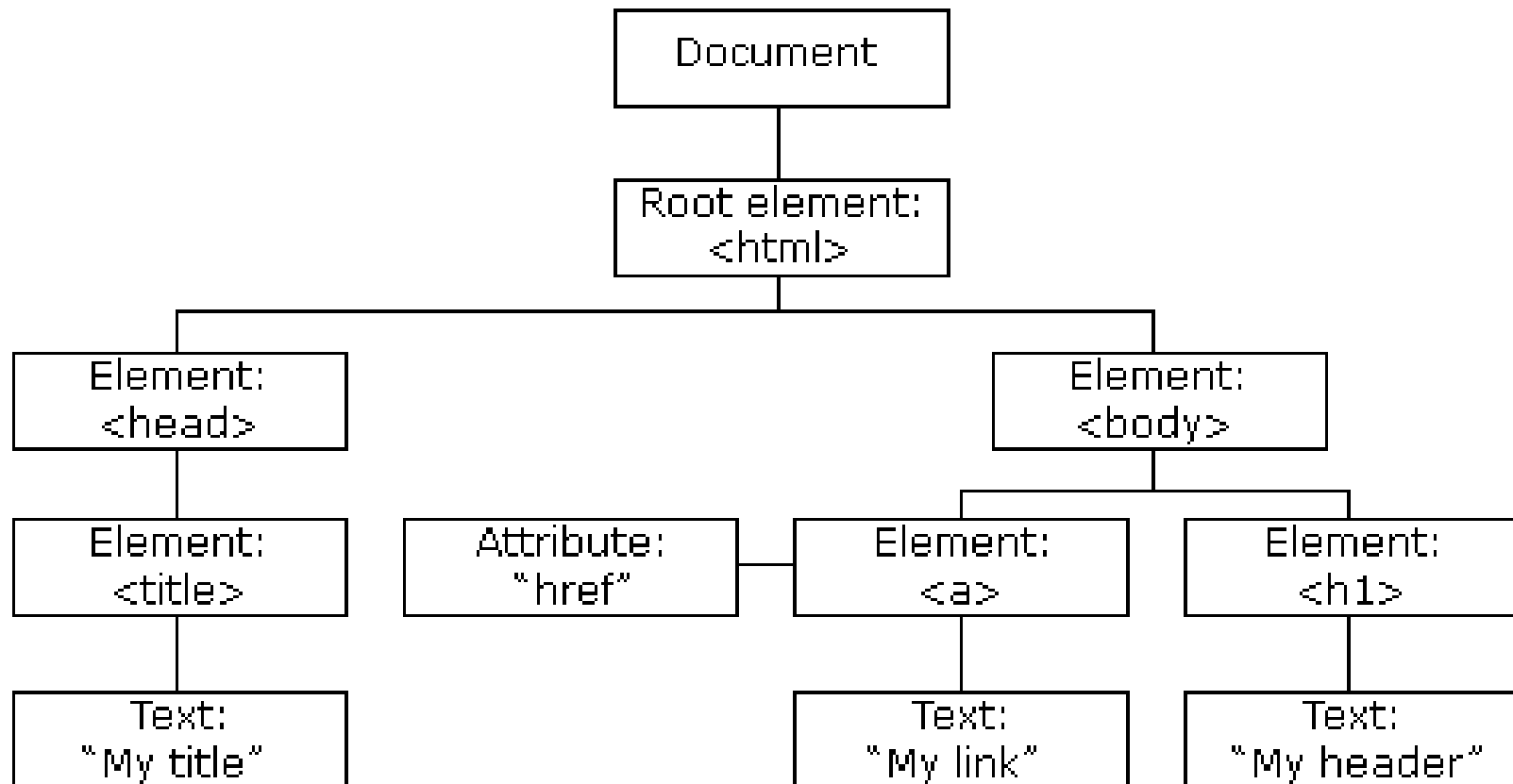
DOM

- 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>
 <p>Ola Mundo!</p>
 </body>
</html>
```
- `</html>` é o nodo raiz (root)
- Todos os outros nodos estão dentro do nodo HTML

# A árvore DOM de nodos



# Acessando nodos - querySelector()

- O método querySelector()
  - Retorna o primeiro elemento de um seletor CSS:
  - `document.querySelector("#menu");`
  - Retorna o elemento com `id="menu"`:

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

```
 let x=document.querySelector("#intro");
 document.write(x.firstChild.nodeValue);
```

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

```
// Lembrando um comando css: p {color:blue}
```

# Acessando nodos - querySelector()

- Neste exemplo o nodo com id="meuTitulo" será capturado pelo comando `document.querySelector("meuTitulo")` e seu conteúdo será jogado na variável `x`. Depois o comando `alert(x.innerHTML)` vai mostrar uma mensagem com o conteúdo 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
 querySelector e mostrado na mensagem alert</h1>
<h1 id="OutroTitulo">Este título não sera capturado</h1>
 <script type="text/javascript">
 let x=document.querySelector("#meuTitulo");
 alert(x.innerHTML);
 </script>
</body>
</html>
```

- Para elementos, `innerHTML` é equivalente a `firstChild.nodeValue`

# Acessando nodos - document.querySelectorAll()

- O método `document.querySelectorAll()`
  - Retorna todos os elementos de um seletor CSS
  - `node.document.querySelectorAll("seletor");`
  - `document.querySelectorAll("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.querySelector('#main').querySelectorAll("p");`



## Acessando nodos - document.querySelectorAll

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

# Acessando nodos - document.querySelectorAll()

- Na primeira linha do script, x recebe uma lista com todos os elementos <p> 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 <p> e </p>

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

# Acessando nodos - `document.querySelectorAll()`

- Exemplos de query...
- `p.aviso` – todos os p com `class="aviso"`
- `P,h1` - todos p e h1
- `#test` quem tiver o id = "test"
- `querySelector("#test")` retorna o primeiro nodo que corresponda a query.

# 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":
  - Similar ao querySelector mas limitado a ids

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

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

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

# 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
  - Similar ao `querySelectorAll` mas limitado a tags
- O código a seguir retorna uma lista de nodos com todos elementos `<p>` que são descendentes do elemento com `id="main"`
  - `document.querySelector('#main').getElementsByTagName("p");`

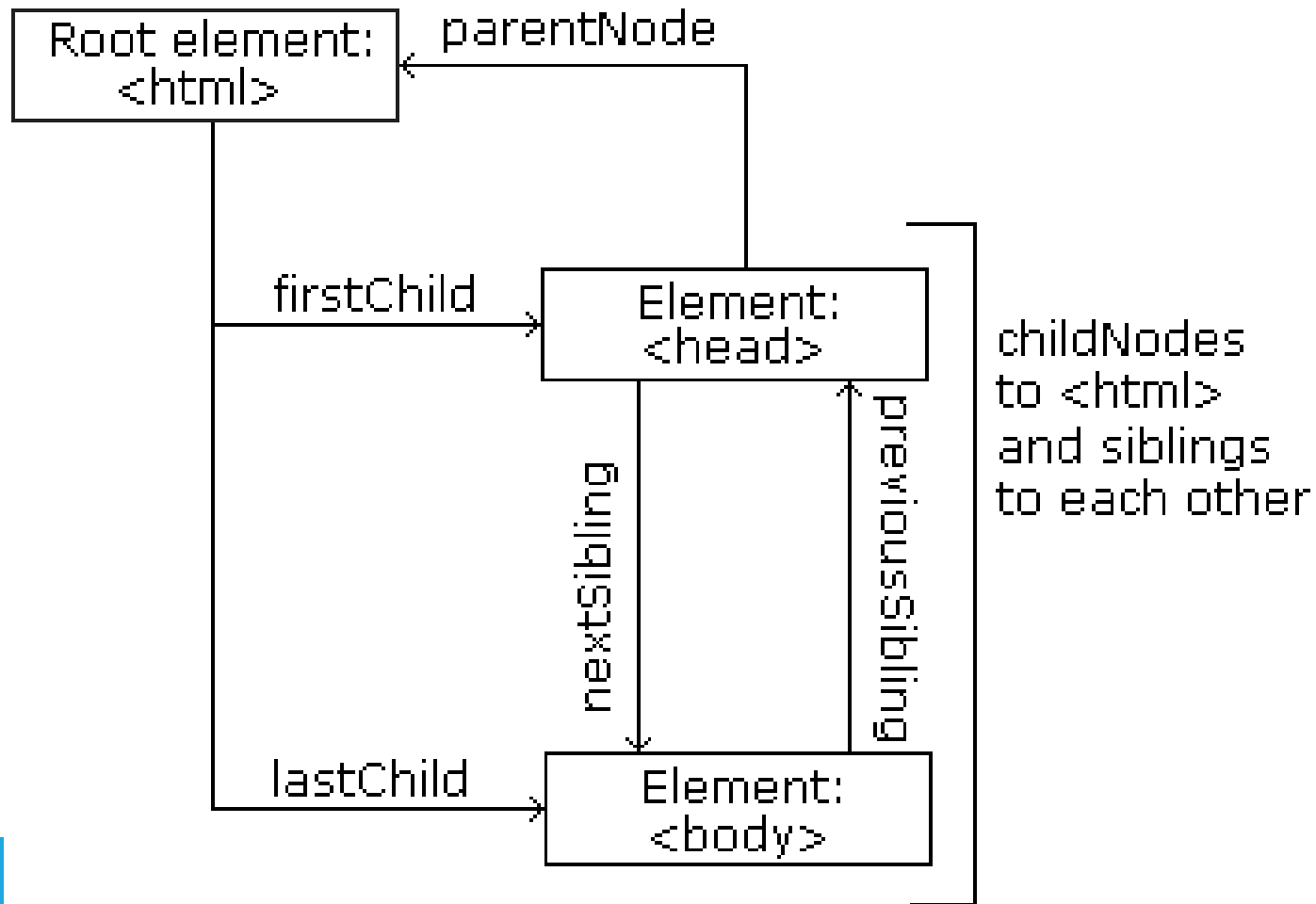
## Acessando nodos – `getElementsByTagName()`

- Similar ao `querySelectorAll` porém limitado a tags
- 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 – Por relacionamentos

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

# Acessando nodos – Por relacionamentos





# Acessando nodos – Por relacionamentos

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

# Acessando nodos – Por relacionamentos

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

# Acessando Propriedades

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

<b>Tipo de nodo</b>	<b>NodeType</b>
<b>Element</b>	<b>1</b>
<b>Attribute</b>	<b>2</b>
<b>Text</b>	<b>3</b>
<b>Comment</b>	<b>8</b>
<b>Document</b>	<b>9</b>

# Acessando Propriedades

```
<html>
<head></head>
<body>
<p id="intro">Alo Mundo!</p>
<hr><hr><hr>

 <script type="text/javascript">
 let x=document.querySelector("#intro");
 //propriedades do elemento body
 document.write(x.parentNode.nodeName + "
");
 document.write(x.parentNode.nodeValue + "
");
 document.write(x.parentNode.nodeType + "
");
 document.write("<HR>");
 // propriedades do elemento P
 document.write(x.nodeName + "
");
 document.write(x.nodeValue + "
");
 document.write(x.nodeType + "
");
 document.write("<HR>");
 // propriedades do texto Alo mundo
 document.write(x.firstChild.nodeName + "
");
 document.write(x.firstChild.nodeValue + "
");
 document.write(x.firstChild.nodeType + "
");
 </script>
</body></html>
```

# Alterando documentos

- Alterando atributos
- ```
<html>
<body>
  <script type="text/javascript">
    document.body.bgColor="yellow";
  </script>
</body>
</html>
```
- Usando o innerHTML (alternativa *insertAdjacentHTML())
- ```
<html>
<body>
<p id="p1">Hello World!</p>
 <script type="text/javascript">
 document.querySelector("#p1").innerHTML="New
text!";
 </script>
</body>
</html>
```

# Alterando documentos

- Usando eventos
- ```
<html>  
<body>  
<input type="button" id="verde" value="Muda cor" />  
<script>  
document.querySelector("#verde").addEventListener("click", ( ) =>  
{document.body.backgroundColor = 'green';} );  
</script>  
</body>  
</html>
```


Alterando documentos

Usando o objeto style //altera prop. CSS

```
<html>
<head></head>
<body>
```



```
<input type="button" id="b1" value="Muda cor" />
```

```
<script type="text/javascript">
document.querySelector("#b1").addEventListener("click",mudaCor );
function mudaCor()
{
document.body.style.backgroundColor="lavender";
}
</script>
</body>
</html>
```

Alterando documentos

- ```
<html><head></head>
<body>
<p id="p1">Hello world!</p>
<input type="button" id="b2" value="Alterar" />
<script type="text/javascript">
document.querySelector("#b2").addEventListener("click", altera);
```
- ```
function altera()
{
document.querySelector("#p1").style.color="blue";
document.querySelector("#p1").style.fontFamily="Arial";
}
</script>
</body>
</html>
```

Operações com atributo

```
let x=document.querySelectorAll("#title")[0];  
let y= x.getAttributeNode("lang");  
let txt=y.nodeValue;
```

```
x=document.querySelectorAll('title')[0];  
x.setAttribute("lang","pt-br");
```

```
btn = document.querySelectorAll("INPUT")[0];  
btn.attributes.removeNamedItem("type");
```

Operações com elementos

- `let node = document.createElement("LI");`
- `let textnode = document.createTextNode("ola");`
- `node.appendChild(textnode);`
- `document.querySelector("#myList").appendChild(node);`

- `let list = document.getElementById("myList");`
- `list.removeChild(list.childNodes[0]);`

```
let x=document.querySelectorAll("title")[0];  
x.removeChild(x.childNodes[0]);
```

```
newTitle=document.createElement("title");  
newText=document.createTextNode("A Notebook");  
newTitle.appendChild(newText);  
newNode.appendChild(newTitle);
```

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).
- **Event listener:** são “alarmes” que detectam quando determinados eventos ocorrem e podem disparar ações quando estes eventos ocorrem.
- A seguir uma lista de eventos i. Também é mostrado os elementos HTML onde estes eventos podem ocorrer e consequentemente onde os manipuladores podem ser colocados.
- **load** - Detecta a carga do documento. Ou seja, quando o usuário acessa a página.
 - Válido para o elemento Body
- **unload** - Detecta quando o usuário sai da página.
 - Válido para o elemento Body
- **change** - 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

- **blur** - 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.
- **focus** - 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.
- **click** - Detecta quando o objeto recebe um Click do Mouse.
 - válido para o Button, Checkbox, Radio, Link, Reset e Submit.
- **mouseover** - Detecta quando o ponteiro do mouse passa por sobre o objeto.
 - válido para Links.
- **select** - Detecta quando o objeto é selecionado.
 - válido para o Text e Textarea.
- **submit** - Detecta quando um botão tipo Submit recebe um click do mouse.
 - válido para o Form.

Eventos

- Os Event Listeners citados anteriormente podem ser associados a determinados elementos HTML. Eles vão cuidar se determinado evento ocorre no elemento onde foram associados
- No exemplo abaixo o listener vai detectar se o usuário muda o texto que esta escrito na caixa de texto “a”. Se isto ocorrer ele dispara uma ação (mostra uma mensagem de alerta).

```
<form name="Text">
  <input type="text" id="a1" size="20" value="" >
</form>
<script>
document.querySelector("#a1").addEventListener("change", mensagem);

function mensagem() {
  alert ('Voce digitou ' + document.querySelector("a1").value)
}
</script>
```


Eventos

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

```
<input type="radio" id="verde" value="1" >Verde  
<input type="radio" id="violeta" value="2"> Violeta  
<input type="radio" id="amarelo" value="3">Amarelo  
<script>  
document.querySelector("#verde").addEventListener("click",  
()=> {document.body.backgroundColor = 'green';});  
document.querySelector("#violeta").addEventListener("click",  
()=> {document.body.backgroundColor = '#violet';});  
document.querySelector("#amarelo").addEventListener("click",  
()=> {document.body.backgroundColor = 'yellow';});  
</script>
```

Eventos

```
<form method="POST" id="form1" action="local.php">  
  Digite um Texto <input type="text" id="teste" value="">  
  Botao Submit <input type="submit" value="Manda p/Server">  
</form>  
<script>  
  document.querySelector("#form1").addEventListener("submit", (e)=>{testa(e)} );  
  
  function testa(e) {  
    if (document.querySelector("#teste").value == "") {  
      alert ("Campo nao Preenchido...Form nao Submetido");  
      e.preventDefault();  
    }  
    else {  
      alert ("Tudo Ok....Form Submetido");  
    }  
  }  
</script>
```

Eventos

mouseover: quando o mouse passa por cima

mouseout: quando o mouse sai de cima

```
<html>
<head>
</head>
<body>
<a href="http://www.google.com" target="_blank">
</a>

<script>
document.querySelector("#b1").addEventListener(mouseover,
()=>{document.querySelector("#b1").src ="b_blue.gif";});
document.querySelector("#b1").addEventListener(mouseout,
()=>{document.querySelector("#b1").src ="b_blue.gif";});

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

Eventos com addEventListener

- `document.querySelector("#teste").addEventListener("click", funcao, true);`
 - Num mesmo elemento podem ser adicionados vários event handlers
 - Click: evento que dispara uma funcao
 - Funcao: função 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.querySelector("#teste").removeEventListener("click", funcao);`
    - Remove o event listener
  - A árvore do DOM não tem hoisting, logo os elementos html referenciados devem estar no código antes de serem referenciados no `addEventListener`. Já as funções tem hoisting então podem ser referenciadas antes de serem definidas

# Mais Eventos

- `document.querySelector("#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.");
 }
}
```

// e.key – retorna o nome da tecla, e.preventDefault() cancela o evento

Quando um event handler chama uma função ele passa para a função 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

# 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></head>
<body>
<form>
Name: <input type="text" id="f1" size="30">

Age: <input type="text" id="age" size="30">
</form>
</body>
<script type="text/javascript">
window.addEventListener("load", setFocus);
function setFocus()
{
document.querySelector("#f1").focus();
}
</script>
</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()
{
let 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">
let t;
    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>
```

Introdução ao JavaScript

Tiago Lopes Telecken

telecken@gmail.com

