Introdução ao JavaScript

Tiago Lopes Telecken

telecken@gmail.com

HTML - DOM

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

DOM

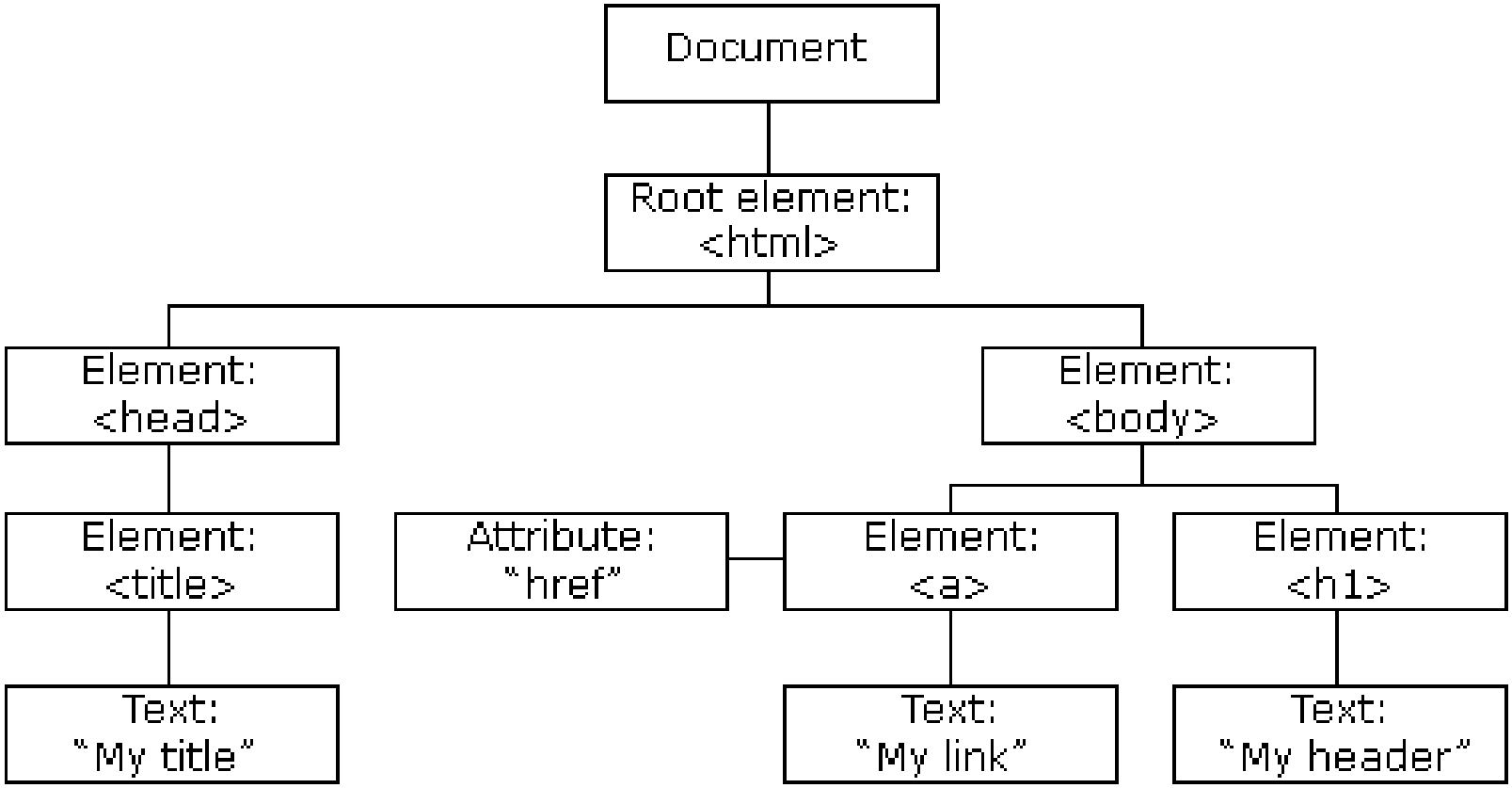
- O DOM transforma todo documento HTML em objetos com propriedades e métodos que podem ser acessados
- Para o DOM, tudo em um documento HTML é um nodo (o nodo é um objeto com propriedades e métodos).
  - O documento inteiro é um nodo
  - Todos elementos HTML são nodos
  - Todos textos nos elementos HTML 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 - 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="alo">1</p>
    <p id="alo2">2</p>
<script>

document.querySelector("#alo").innerHTML  =" Alo Mundo!";

    </script></body></html>
// Lembrando um comando css:   p {color:blue}
```

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 um array de nodos
- Selecionar todos nodos <p>
  - x=document.querySelectorAll("p");
- Acessar o segundo p do array
  - y=x[1];
- Tamanho do array
  - x.length

Acessando nodos - document.querySelectorAll()

- 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>
<h2>primeiro</h2>
<h2>segundo</h2>
<hr>
<div></div>
<script>
    let resposta = document.querySelector("div");
    let lista=document.querySelectorAll("h2");
    for (let i=0;i<lista.length;i++)
    {
          resposta.innerHTML+= `${lista[i].innerHTML} <br>`;
    }
</script></body></html>
```

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>
    <div></div>
    <script type="text/javascript">
          let x=document.getElementById("intro");
          document.querySelector("div").innerHTML=
          `${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.querySelectorAll('#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

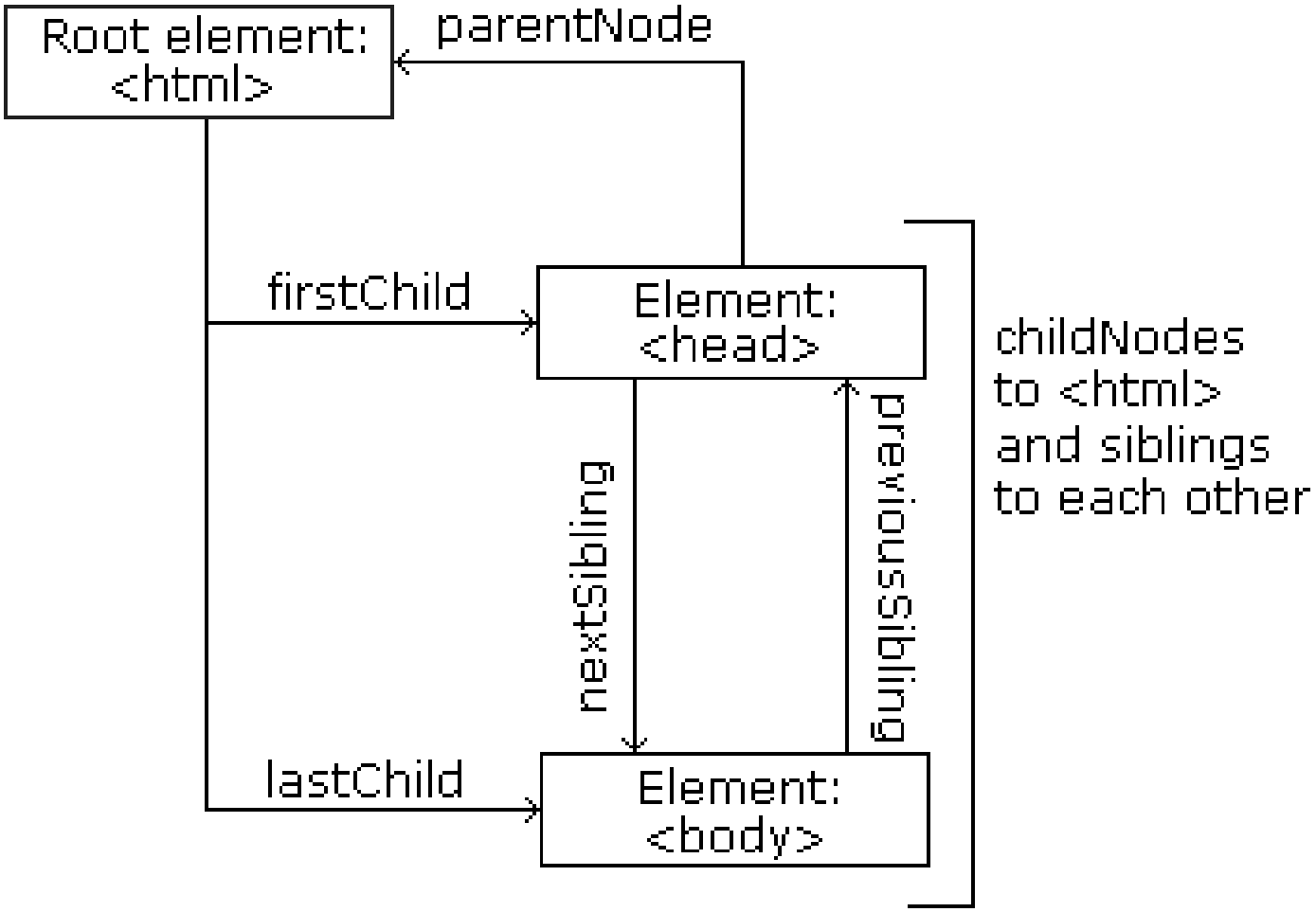
Acessando nodos – Por atalhos

- document.documentElement – retorna o elemento raiz <html>
- document.body – retorna o elemento <body>

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 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**

Acessando Propriedades

- nodeType
- Retorna o tipo do nodo

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

Acessando Propriedades e Relacionamentos

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

<p id="intro">Document Object Model</p>
<hr><hr><hr><br>
<div id="resposta"></div>
<script>
let x=document.querySelector("#intro");

document.querySelector("#resposta").innerHTML=`

<HR>Propridades do elemento P <br><br>
${x.nodeName} <br>
${x.nodeValue} <br>
${x.nodeType} <br>
<HR>Propridades do texto <br><br>
${x.firstChild.nodeName} <br>
${x.firstChild.nodeValue} <br>
${x.firstChild.nodeType} <br>`;
Propridades do elemento body<br><br>
${x.parentNode.nodeName}<br>
${x.parentNode.nodeValue}<br>
${x.parentNode.nodeType} <br>

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

Alterando documentos

- Usando o innerHTML
- ```
  <html>
  <body>
  Hello World!
  <script>

      document.querySelector("#p1").innerHTML="Alo JS!";

  </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">
function mudaCor()
{
document.body.style.backgroundColor="lavender";
}

document.querySelector("#b1").addEventListener("click",mudaCor );
</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">
function altera()
{
document.querySelector("#p1").style.color="blue";
document.querySelector("#p1").style.fontFamily="Arial";
}
document.querySelector("#b2").addEventListener("click", altera);
</script>
</body>
</html>
```

Operações com atributo

```
tagTitulo=document.querySelector("title");

// capturar o valor de um atributo
atributoLang= tagTitulo.getAttributeNode("lang").nodeValue;

// Alterar/inserir o valor de um atributo
tagTitulo.setAttribute("lang","pt-br");

// excluir um atributo
tagTitulo.attributes.removeNamedItem("lang");
```

Operações com elementos e textos

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

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

Eventos e ouvintes 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. Também é mostrado os elementos HTML onde estes eventos podem ocorrer e consequentemente onde os ouvintes 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>
function mensagem() {
  alert    ('Voce digitou ' + document.querySelector("#a1").value)
}
document.querySelector("#a1").addEventListener("change", mensagem);
</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.style.backgroundColor = 'green';"});
document.querySelector("#violeta").addEventListener("click",
()=> {document.body.style.backgroundColor = '#violet';"});
document.querySelector("#amarelo").addEventListener("click",
()=> {document.body.style.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">
<img border="0" alt="Visite o Google!" src="b_pink.gif" id="b1" /></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: 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

  oi
- 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 onome da tecla, e.preventDefault() cancela o evento

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

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"><br />
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