PUC Minas

Tecnologias Web

Linguagem Javascript
Document Object Model (DOM)

Prof. Rommel Vieira Carneiro

Javascript – Tópicos

- Document Object Model (DOM)
- Objetos do DOM
- Estrutura do Documento
- Navegação
- Manipulação

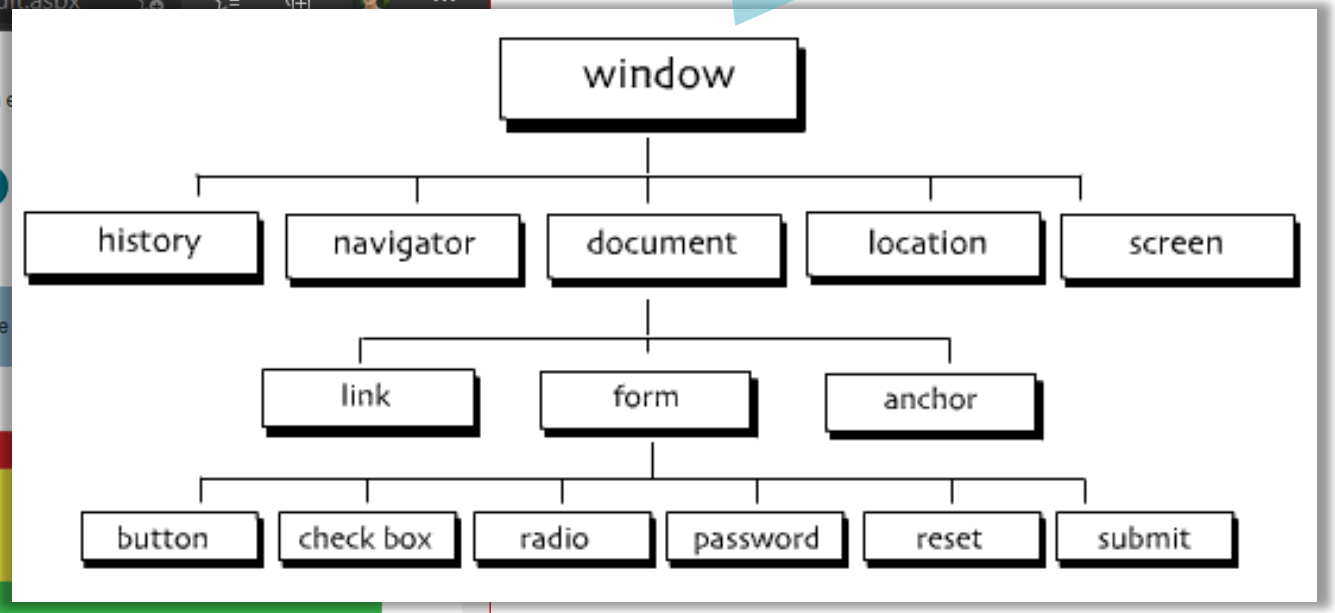
Document Object Model (DOM)

Document Object Model (DOM)

API (*Application Programming Interface*) orientada a objetos para manipulação de estruturas do navegador e do documento Web (conteúdo, estrutura e estilos)

O **DOM** é um padrão **W3C**, independente de plataforma e linguagem.

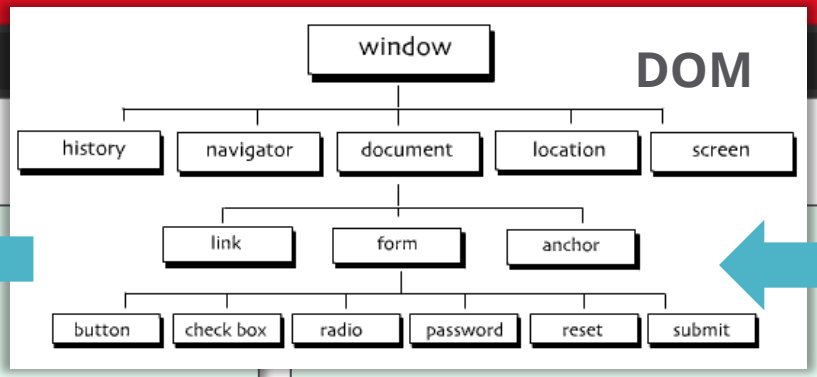
A figura mostra a hierarquia dos objetos do DOM



Document Object Model (DOM)

about:blank

about:blank



Página HTML

```
<!DOCTYPE html>
<html lang="pt-br">
<head></head>
<body>
  <h1> Título </h1>
  <button id="btnHello">
      Hello</button>
   </body>
  <script src="script.js"></script>
</html>
```

JavaScript

```
var elem = document.getElementById('btnHello');
elem.addEventListener('click', function () {
        alert("Hello World!");
    }, false);
```

Document Object Model (DOM) – Objetos

Objetos do Modelo de Objetos (DOM)

- **Window** – representa a janela do navegador
- **Navigator** – contém informações sobre o navegador
- **Screen** – contém informações sobre a tela do cliente
- **History** – contém os sites visitados
- **Location** – contém informações sobre a URL atual
- **Document** – representa um documento HTML e pode ser usado para acessar todos os outros elementos:

  Anchor, Area, Base, Body, Button, Event, Form, Frame, Frameset, IFrame, Image, Input Button, Input Checkbox, Input File, Input Hidden, Input Password, Input Radio, Input Reset, Input Submit, Input Text, Link, Meta, Object, Option, Select, Style, Table, TableCell, TableRow, Textarea

DOM – Eventos

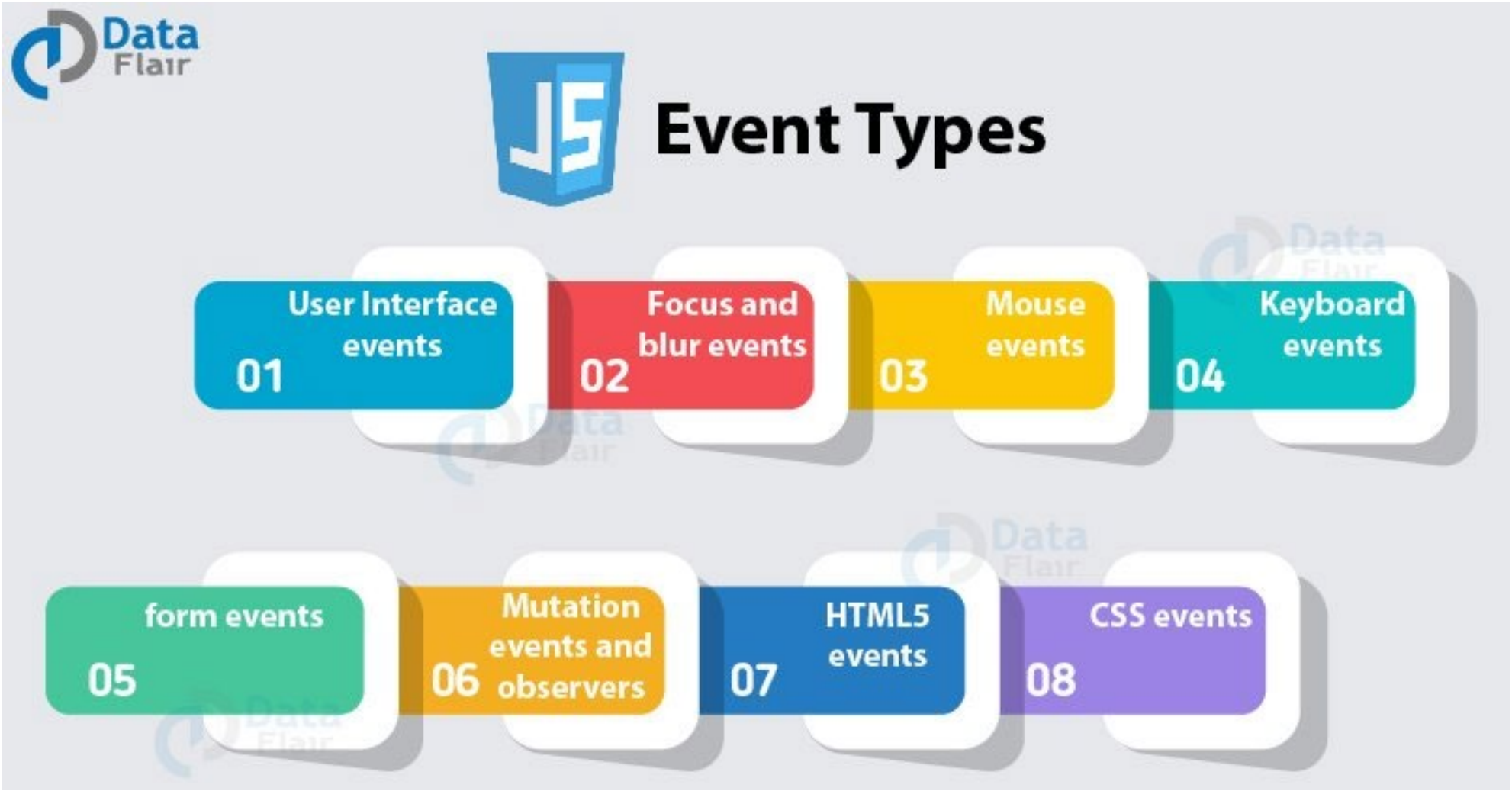
Tratar os eventos em uma página pela inclusão de listeners no código JavaScript separa códigos HTML e JavaScript e torna o código mais legível.

Veja o exemplo:

```
<button id="btnHello">Hello</button>

<script>
    var elem = document.getElementById('btnHello');
    elem.addEventListener('click', function () {
        alert("Hello World!");
    });
</script>
```

DOM – Eventos – Tipos de Eventos



Fonte: JavaScript Event Types

DOM – Eventos

Evento	Quando ocorre...	Controle
click	Usuário clica no link ou element de formulário	onclick
change	Usuário altera o valor de elemento de texto, textarea ou seleção	onchange
focus	Usuário posiciona o foco em um element de formulário	onfocus
blur	Usuário tira o foco de um element de formulário	onblur
mouseover	Usuário move o mouse sobre um elemento	onmouseover
mouseout	Usuário move o mouse para fora de um elemento	onmouseout
select	Usuário seleciona um campo de entrada de formulários	onselect
submit	Usuário submete um formulário	onsubmit
resize	Usuário redimensiona a janela do browser	onresize
load	O elemento é carregada no browser (window, body, frame, img, link)	onload
unload	Usuário sai da página	onunload

Esses são alguns exemplos de eventos. Existem outros.

DOM – Eventos – Objeto Evento

Ao disparar a função de tratamento de um evento, o DOM passa, por parâmetro, um apontador para um objeto que representa o evento à função de tratamento.

No exemplo abaixo, utilizamos o objeto evento passado como parâmetro para o tratamento do evento e cancelamos o processamento do formulário para fazer um tratamento alternativo

```
<script>
   // Inclui listener para evento submit passando o evento como parâmetro
   document.getElementById("formX").addEventListener("submit",
        function(event) {
            event.preventDefault()

            // Evita o envio do form ao servidor e processa de forma diferente
            ...
        });
</script>
```

DOM – Eventos – Objeto Evento

Principais propriedades e métodos do objeto evento:

Propriedade/Método	Descrição
target	Uma referência para o elemento que dispara o evento. Ex: em um clique, referencia o elemento clicado
type	Especifica o tipo do evento disparado. Ex: em um clique, traz o valor “click”
timeStamp	Especifica o momento em que o evento foi disparado em milissegundos.
preventDefault ()	Permite cancelar o comportamento padrão Ex: cancelar submissão do formulário no evento onsubmit

JavaScript – Eventos

Exemplo

```
<!DOCTYPE html>
<html>
<head>
    <title>Exemplo DOM - Eventos</title>
</head>
    <body>
      <p onmouseover="this.style.color='red';"
         onmouseout="this.style.color='black';" >
         Olá Mundo
      </p>
    </body>
</html>
```

No exemplo acima, ao passar o mouse sobre o parágrafo, a cor do texto é alternada para vermelho.

A palavra-chave **this**, significa o próprio elemento onde o evento se encontra.

Estrutura do Documento

No DOM, tudo é **nodo**:

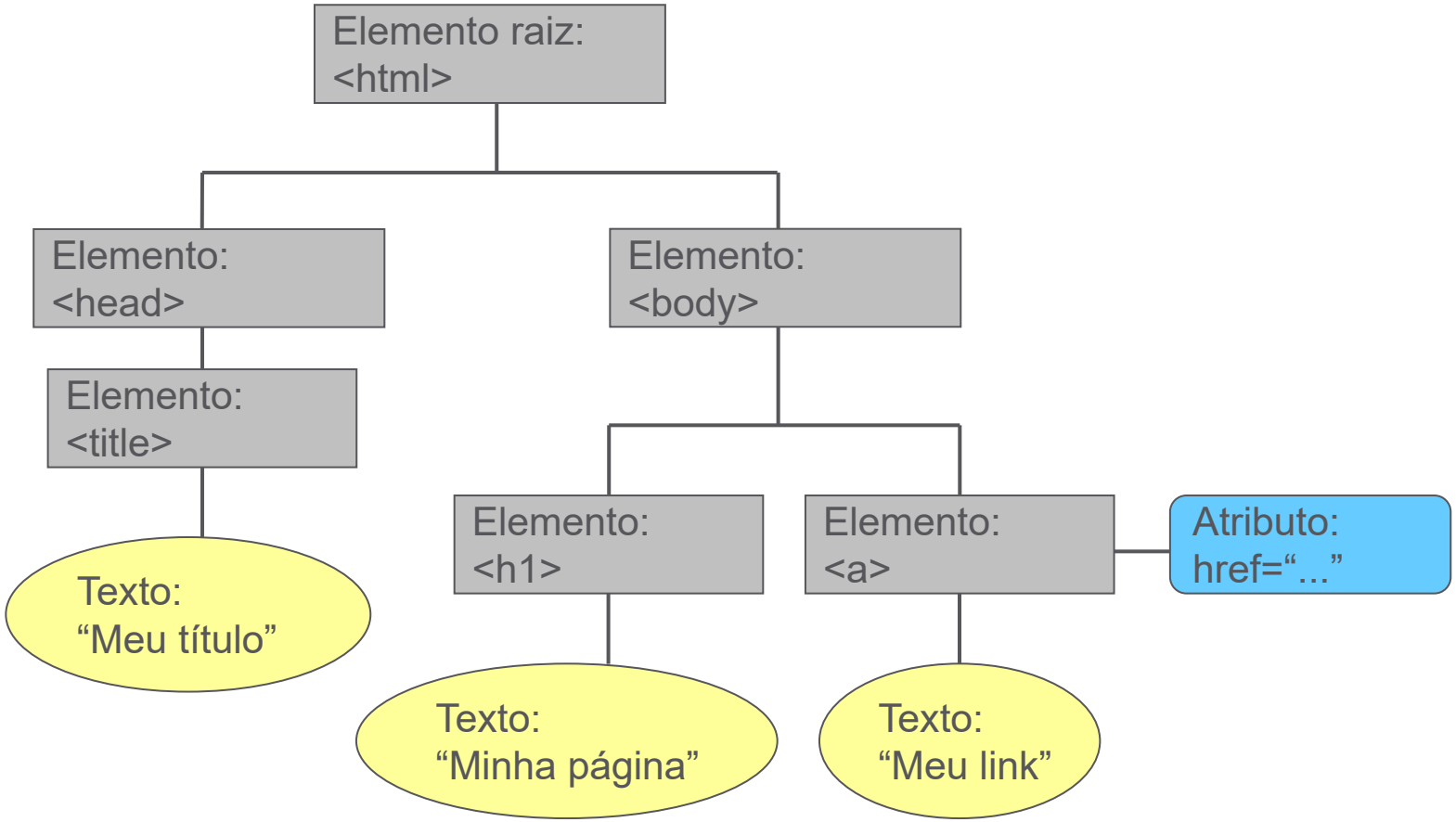
- O documento (inteiro) é um nodo (document node)
- Todo elemento é um nodo (element node)
- O texto dos elementos são nodos (text node)
- Cada atributo é um nodo (attribute node)
- Os comentários são nodos (comment node)

A relação entre os nodos é semelhante a uma árvore

- Eles possuem elementos pai (parent) e filho (child)
- A raiz (document node) não possui pai, nem as folhas possuem filhos

Estrutura do Documento

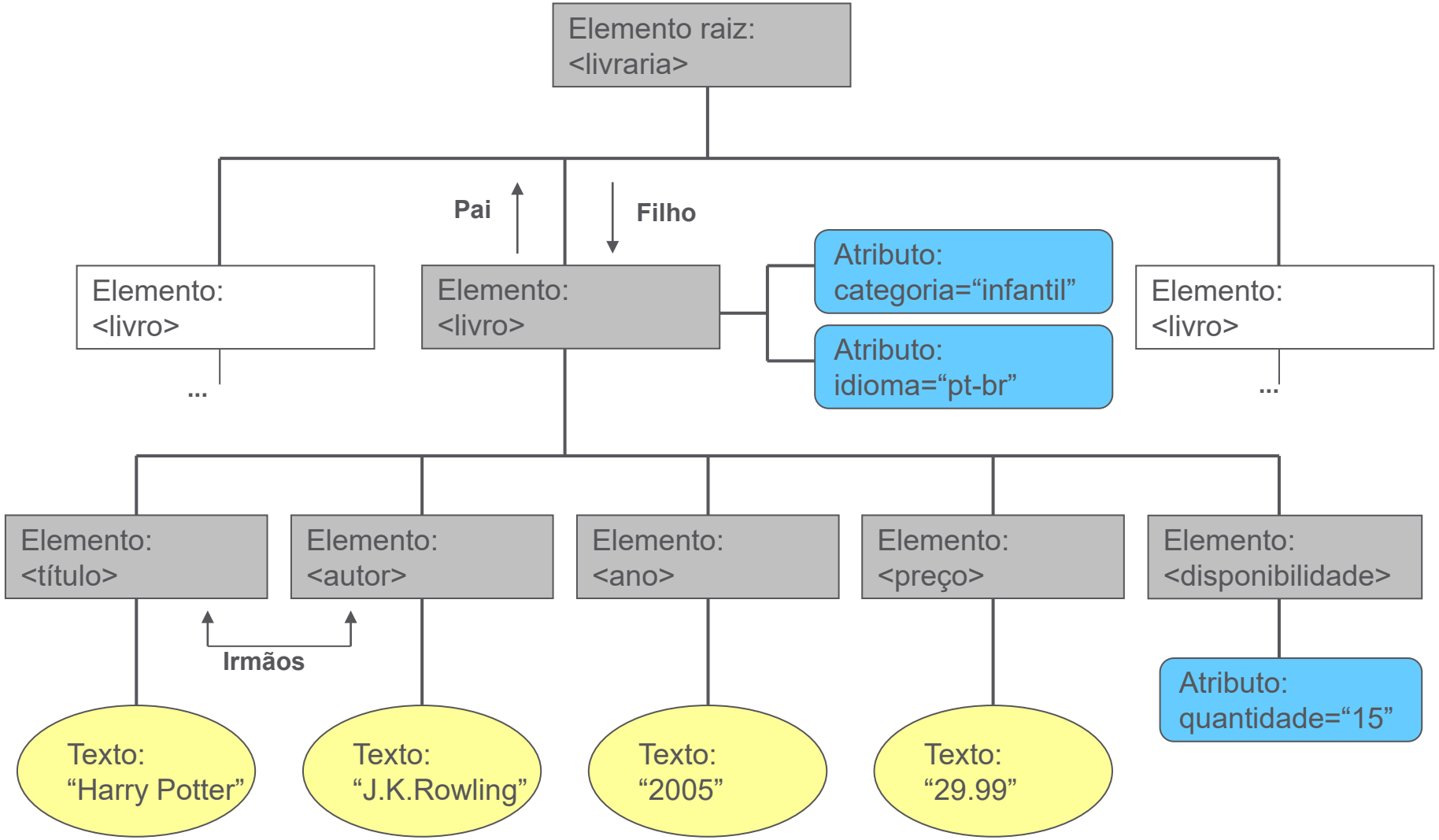
O DOM oferece acesso a documentos HTML ou XML como uma árvore, onde todo elemento, seus atributos, e seu conteúdo são identificados e tratos como **nodos**.



Estrutura do Documento

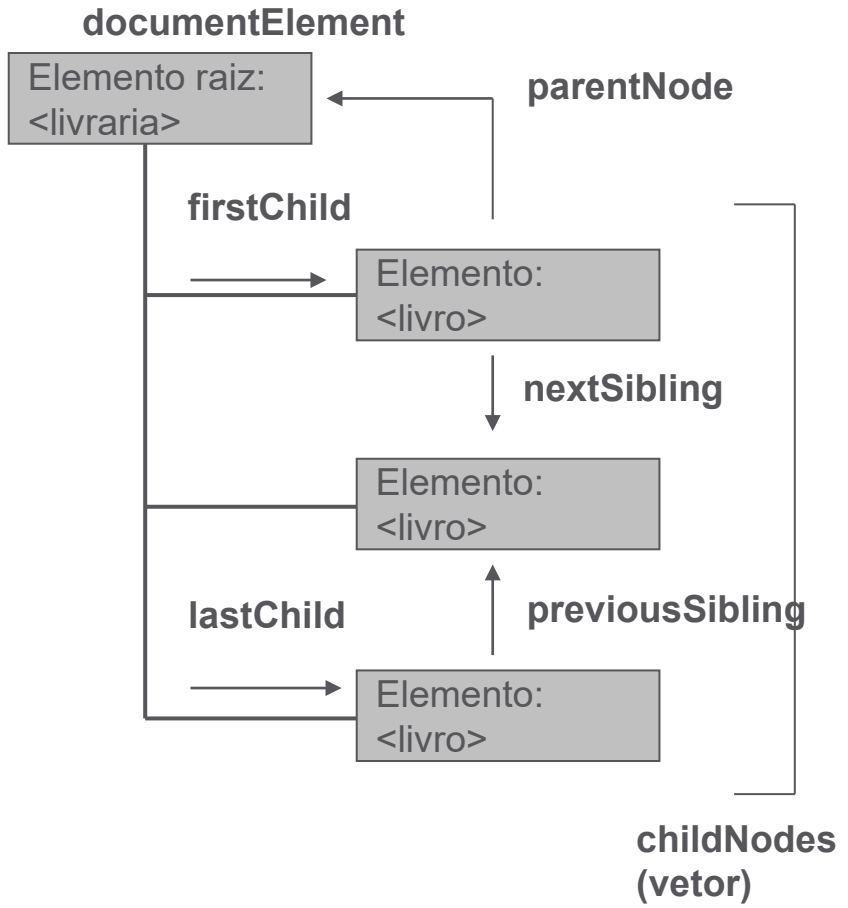
```
<?xml version="1.0" encoding="iso-8859-1"?>
<livraria>
     <livro categoria="culinária" idioma="en">
        <título>Everyday Italian</título>
        <autor>Giada de Laurentis</autor>
        <ano>2005</ano>
        <preço>30.00</preço>
        <disponibilidade quantidade="3"/>
    </livro>
    <livro categoria="infantil" idioma="pt-br">
        <título>Harry Potter</título>
        <autor>J.K.Rowling</autor>
        <ano>2005</ano>
        <preço>29.99</preço>
        <disponibilidade quantidade="15"/>
    </livro>
</livraria>
```

Estrutura do Documento



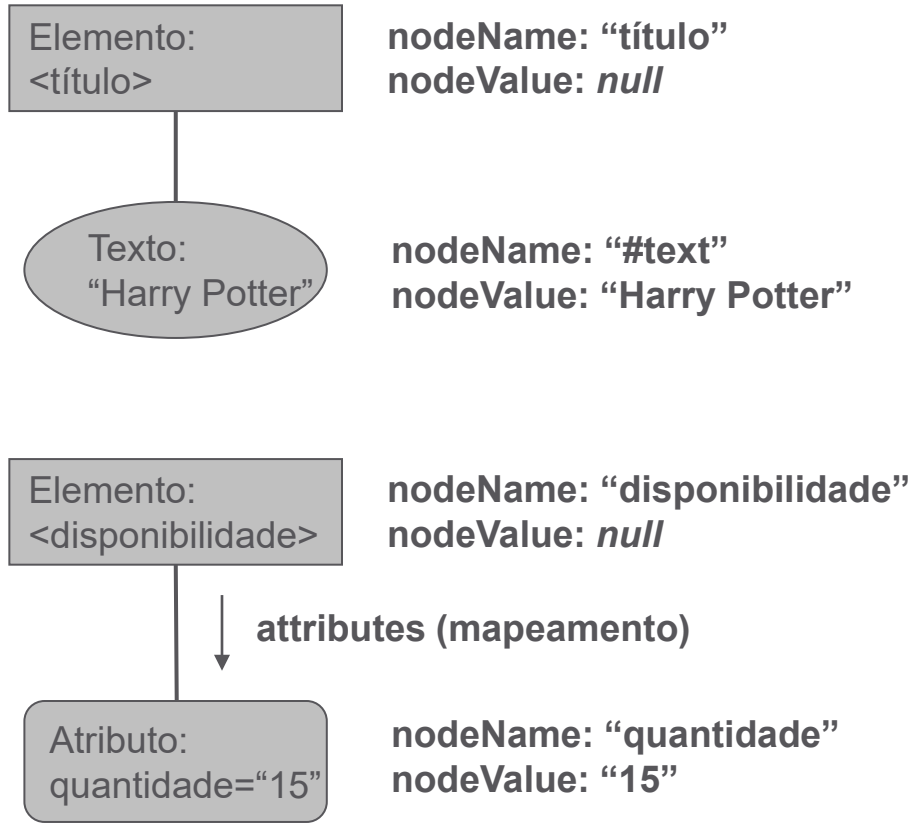
Estrutura do Documento

- O nodo de mais alto nível é chamado de raiz
- Cada nodo, exceto a raiz, tem exatamente um pai
- Um nodo pode ter qualquer número de filhos
- Uma folha é um nodo sem filhos
- Irmãos são os nodos com o mesmo pai



Estrutura do Documento

- O texto de um nodo é também um nodo: o nodo texto
- O texto do element fica armazenado como valor do nodo texto
- Os atributos tambémsão nodos, do tipo atributo
- Atributos não possuem pai
- Atributos possuem um nome e um valor



Estrutura do Documento

Nodo raiz: **documentElement**

Navegação a partir de um nodo qualquer

- Nodo pai: **parentNode**
- Primeiro filho: **firstChild**
- Último filho: **lastChild**
- Próximo irmão: **nextSibling**
- Irmão anterior: **previousSibling**
- Lista de filhos (como um vetor): **childNodes**

Estrutura do Documento

A propriedade **nodeName** retorna o nome do nodo (em caixa alta)

- Se for um elemento, retorna o nome do elemento
- Se for um atributo, retorna o nome do atributo
- Se for texto, retorna #text
- Se for o nodo documento, retorna #document

A propriedade **nodeValue** retorna o valor do nodo

- Se for um elemento ou nodo documento, não funciona
- Se for um texto, retorna o texto
- Se for um atributo, retorna seu valor

Estrutura do Documento

A propriedade **nodeType** retorna o tipo do nodo, sendo:

- Elemento = 1
- Atributo = 2
- Texto = 3
- ...
- Documento = 9

NodeType	Named Constant
1	ELEMENT_NODE
2	ATTRIBUTE_NODE
3	TEXT_NODE
4	CDATA_SECTION_NODE
5	ENTITY_REFERENCE_NODE
6	ENTITY_NODE
7	PROCESSING_INSTRUCTION_NODE
8	COMMENT_NODE
9	DOCUMENT_NODE
10	DOCUMENT_TYPE_NODE
11	DOCUMENT_FRAGMENT_NODE
12	NOTATION_NODE

Fonte: http://www.w3schools.com/dom/dom_nodetype.asp

DOM – Navegação

O método **getElementsByTagName** ("elemento") ignora a estrutura do documento e busca elementos com o nome especificado. Ao encontrar, retorna um vetor de nodos.

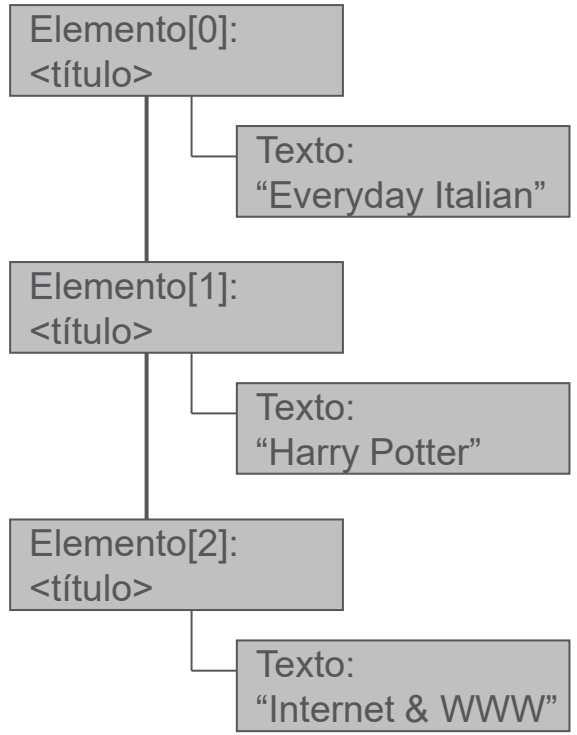
Exemplo:

```
var x = xmlDoc.getElementsByTagName("livro");
for (var i = 0; i < x.length; i++)
{
   // faz algo com cada elemento <livro>
}
```

DOM – Navegação

- Uma lista de nodos é retornada pela propriedade **childNodes** ou pelo método **getElementsByTagName** ()

- O tamanho da lista é determinado com a propriedade **length**



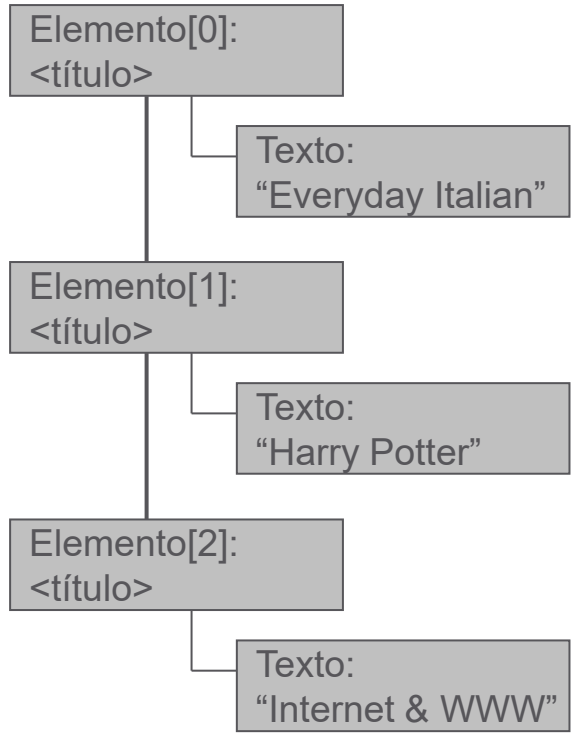
Exemplo:

```
getElementsByTagName("titulo")[0].childNodes[0].nodeValue
// Retorna: Everyday Italian
```

DOM – Navegação

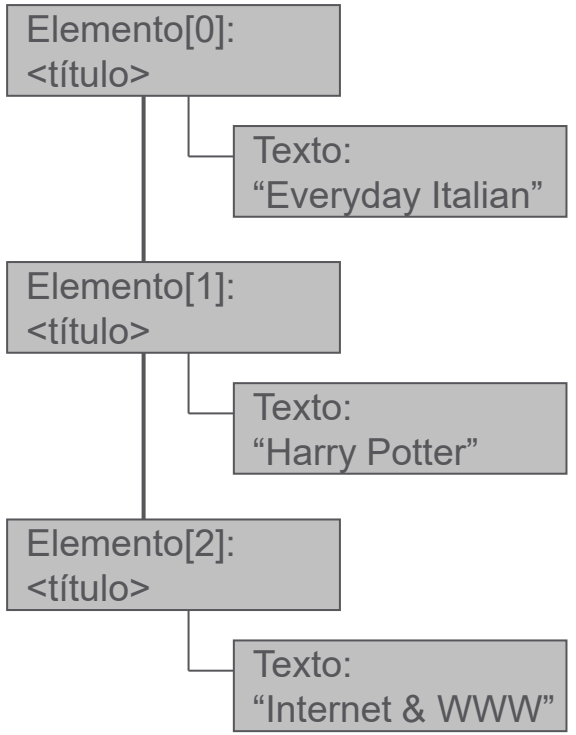
- Alternativas:
  - **getElementsByTagName** ()
    retorna uma lista de elementos
  - **getElementById** ()
    retorna um único elemento

    Exemplo: ....



DOM – Navegação

- Método: **getElementsByTagName** ()
  Recuperação de elementos pelo nome

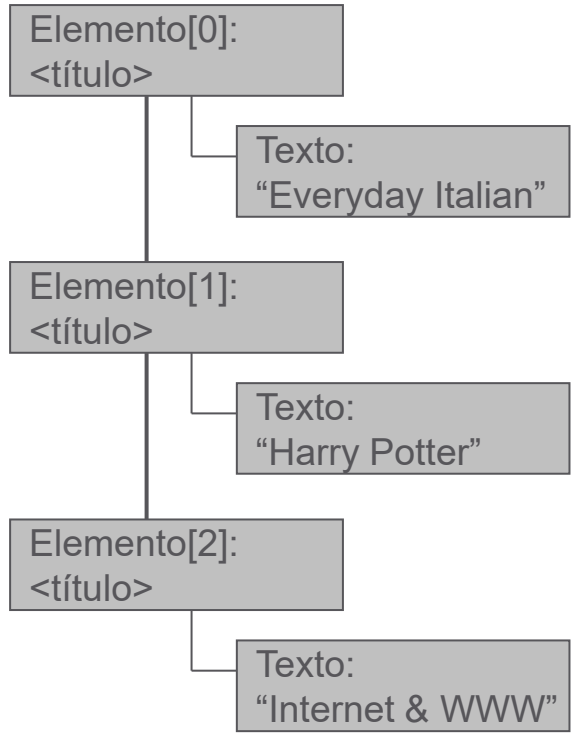


Exemplo

```
xmlDoc=loadXMLDoc("livros.xml");
var x = xmlDoc.getElementsByTagName('title');
for (i = 0; i < x.length; i++) {
    document.write(x[i].childNodes[0].nodeValue);
    document.write("<br />");
}
```

DOM – Navegação

- Método: **getAttribute** ()
  Recupera atributos de um elemento pelo nome do atributo



Exemplo

```
xmlDoc=loadXMLDoc("livros.xml");
var x = xmlDoc.getElementsByTagName('livro');
for (i = 0; i < x.length; i++) {
    document.write(x[i].getAttribute('categoria'));
    document.write("<br />");
}
```

DOM – Manipulação

- Usar o método **setAttribute** ("nome", "valor")

```
xmlDoc=loadXMLDoc("livros.xml");
var  x=xmlDoc.getElementsByTagName("livro");
for(i = 0; i < x.length; i++) {
    x.item(i).setAttribute("edição","primeira");
}
```

- Ou o método **createAttribute** ("nome")

```
xmlDoc=loadXMLDoc("livros.xml");
var x = xmlDoc.getElementsByTagName('livro');
var newatt;
for (i = 0; i < x.length; i++) {
     newatt=xmlDoc.createAttribute("edição");
     newatt.value="primeira";
     x[i].setAttributeNode(newatt);
}
```

DOM – Manipulação

- Remover um elemento – removeChild (nodo)
  - Requer a especificação do nodo a ser removido
- Remover um nodo atributo – removeAttributeNode (nodo)
- Remover o texto de um elemento – deleteData (início, tamanho)
  - Requer a especificação de onde começar a remover (0-based) e quantos caracteres
  - O nodo deve ser texto
- Remover diretamente um atributo – removeAttribute (nome)

DOM – Manipulação

- Substituir um nodo – replaceChild (novo_nodo, nodo_velho)
- Substituir um texto – replaceData (inicio, tamanho, string)
- Criar um elemento – createElement (nome)
- Criar um atributo – createAttribute (nome)
  - A propriedade value do novo atributo permite definir seu valor
- Criar um texto – createTextNode (texto)
- Criar um comentário – createComment (comentário)

DOM – Manipulação

- Adicionar um nodo ao fim da lista – appendChild (nodo)
- Adicionar antes de um nodo especificado – insertBefore (novo_nodo, nodo_especificado)
- Adicionar um atributo a um elemento – setAttribute (nome, valor)
- Adicionar texto a nodo texto – insertData (inicio, texto)
- A clonagem é feita com o método cloneNode (modo)
  - modo pode ser true ou false, dependendo do interesse em se copiar também todos os atributos e nodos filho

Obrigado!

Prof. Rommel Vieira Carneiro