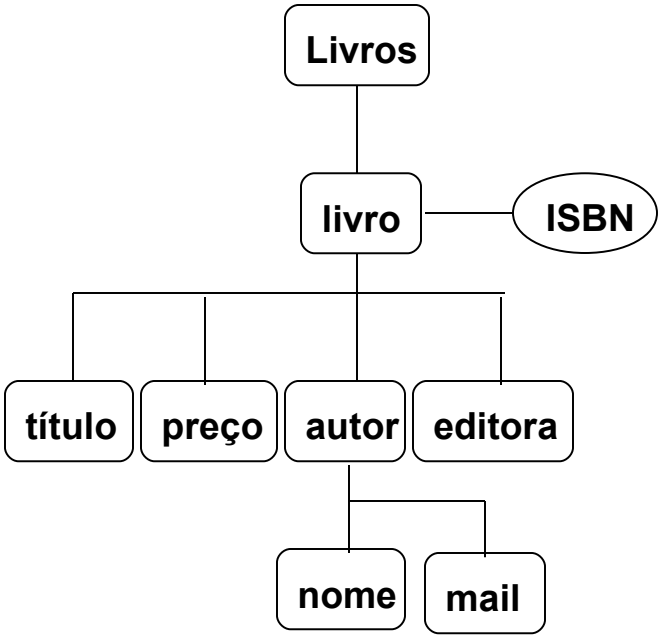
DOM (*Document Object Model*)

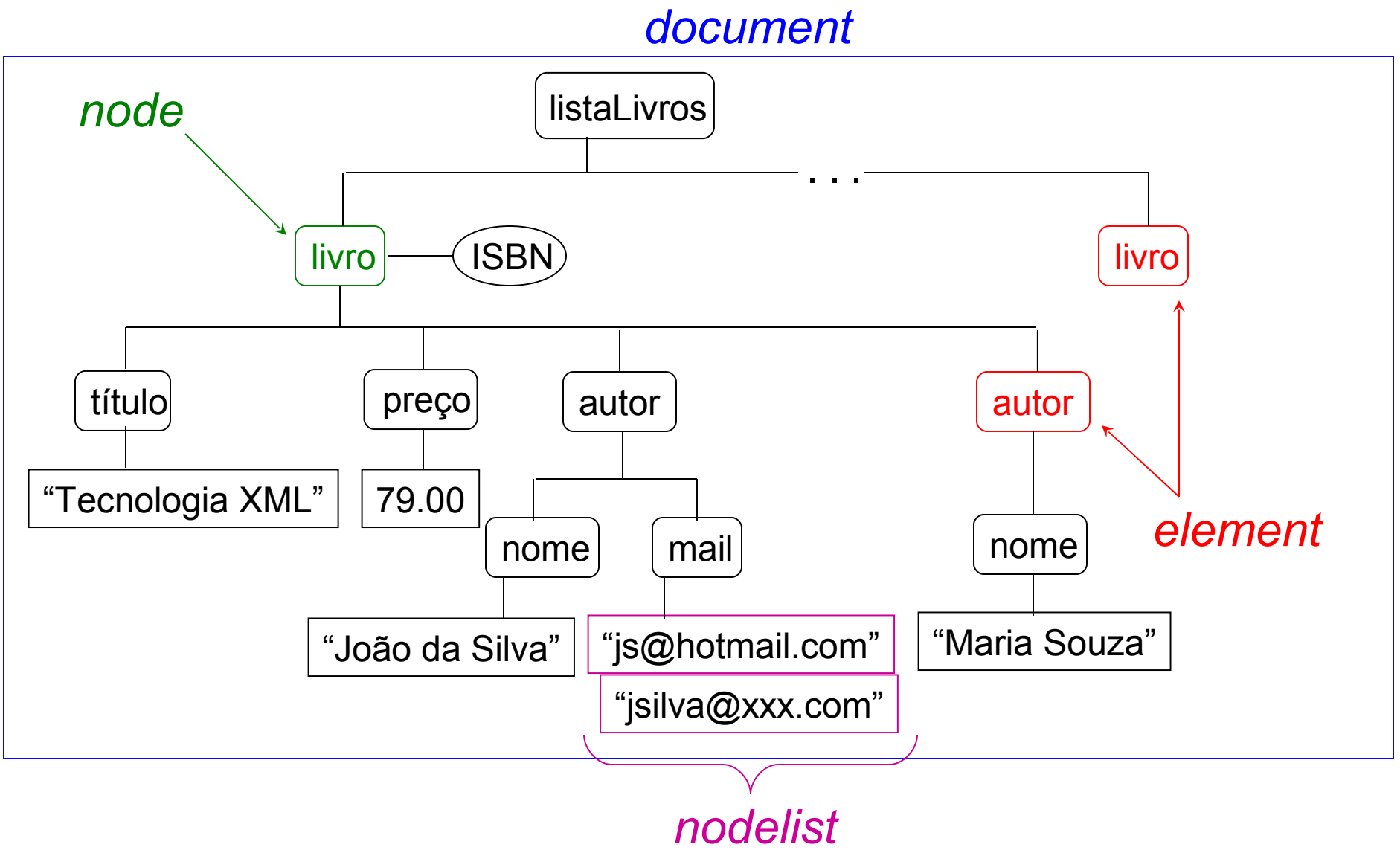
- Modelo de dados para XML
  - estrutura hierárquica (árvore)
  - métodos de acesso (API DOM)
    - principais classes de objetos
      - *document*, *node*, *nodelist* e *element*
    - execução de consultas e atualizações de dados
- *Parsers* DOM
  - validam um doc XML
  - geram um objeto *document*

Esquema DOM

```
<?xml version =“1.0”>
<Livros>
<livro ISBN=“112”>
  <titulo>Tecnologia XML</titulo>
  <preco>79.00</preco>
  <autor>
    <nome>João da Silva</nome>
    <mail>js@hotmail.com</mail>
    <mail>jsilva@xxx.com</mail>
  </autor>
  <autor>
    <nome>Maria Souza</nome>
  </autor>
  <editora>Campus</editora>
  ...
</livro>
...
</Livros>
```



Objetos do Modelo DOM



Principais Métodos

document

Método	Resultado
documentElement	Element
getElementByTagName(String)	NodeList
createTextNode(String)	String
createComment(String)	Comment
createElement(String)	Element

Principais Métodos

node

Método	Resultado
nodeName	String
nodeValue	String
nodeType	short
parentNode	Node
childNodes	NodeList
firstChild	Node
lastChild	Node
previousSibling	Node
nextSibling	Node
insertBefore(Node novo, Node ref)	Node
replaceChild(Node novo, Node antigo)	Node
removeChild(Node)	Node
hasChildNode	boolean

Principais Métodos

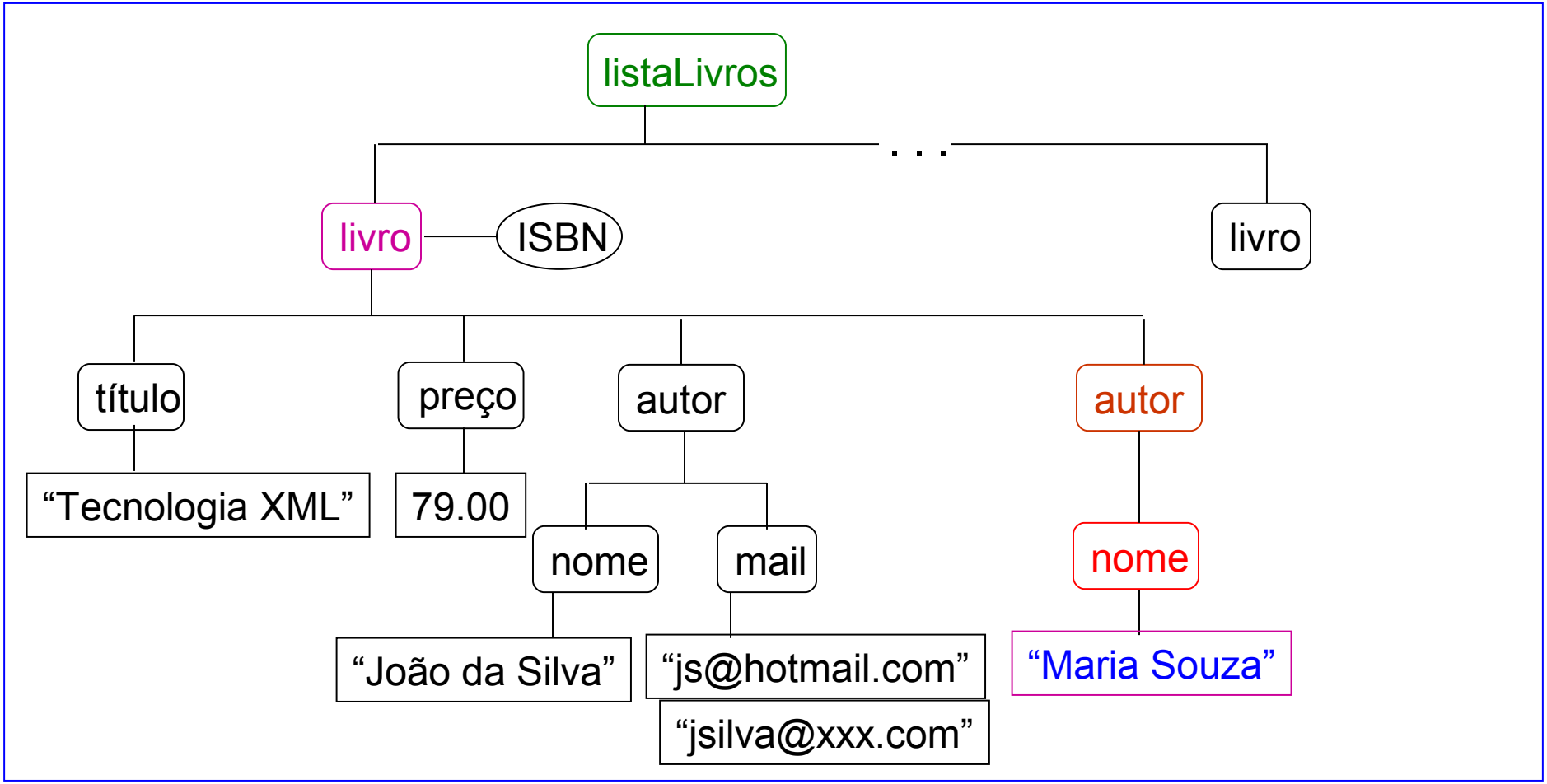
element

Método	Resultado
tagName	String
getAttribute(String)	String
setAttribute(String nome, String valor)	Attr
getAttributeNode(String)	Attr
removeAttributeNode(String)	Attr
getElementsByTagName	NodeList

nodeList

Método	Resultado
Length	int
item(int)	Node

Exemplo de Navegação em DOM



doc.documentElement.childNodes.item(0).getElementsByTagName("autor").item(1).firstChild.firstChild.data

objeto DOM	nodo raiz	lista de livros	1º livro	lista de autores	2º autor	1º nodo filho: nome	1º nodo filho: conteúdo de nome	texto

DOM – Exemplo (*JavaScript*)

```
var doc, raiz, livro1, autores, autor2;
doc = new ActiveXObject("Microsoft.XMLDOM");
doc.load("livros.xml");
if (doc.parseError != 0) ...;
else
{
   raiz = doc.documentElement;
   /* busca o primeiro livro (primeiro nodo filho) */
   livro1 = raiz.childNodes.item(0);
   /* busca a lista de autores do primeiro livro */
   autores = livro1.getElementsbyTagName("autor");
   /* busca o segundo autor */
   autor2 = autores.item(1);
   /* escreve o nome do autor - primeiro nodo filho */
   document.write("Nome do segundo autor: " +
     autor.childNodes.item(0).data);
}
```

XSL (*XML Style sheet Language*)

- *Style sheet* (folha de estilos)
  - regras para formatação da apresentação de dados (regras para definição de estilo)
  - *CSS (Cascading Style Sheet)*
    - padrão estendido pela W3C para formatar dados XML em *browsers Web*
    - um arquivo .css pode ser referenciado em um doc XML
    - características de formatação para *tags* XML fonte + cor + cor *background* + margens + posicionamento + listas + tabelas + ... +

XSL (*XML Style sheet Language*)

- XSL
  - estende as funcionalidades do CSS
    - formatação de apresentação (como CSS)
    - transformação do conteúdo do documento XML (XSLT)
      - indicação de que dados serão exibidos ou descartados
      - inserção de novos conteúdos
      - conversão XML→HTML, XML→XML, XML→texto puro, ...

Documento XSL

- Define uma folha de estilo
- Sintaxe XML
- Referenciado em um doc XML

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="estilo.xsl"?>
...
```

- Processador XSL
  - programa que valida e executa as regras definidas em um doc XSL
  - alguns *browsers Web* processam docs XSL

Estrutura de um Doc XSL(T)

```
<stylesheet xmlns = "http://www.w3.org/XSL/Transform/1.0">
</template match = "/livro/autor">
...
</template>
</stylesheet>
```

elemento raiz

namespace default
(DTD da W3C com instruções XSL)

padrão: indica o elemento ou atributo para o qual a regra se aplica
(expressão XPath)

regra de formatação

XSL Transformation - XSLT

- Sintaxe para transformação de dados
- Algumas instruções
  - **<xsl:apply-templates [select="..."]>**
    - processa (recursivamente) todos os sub-nodos do nodo corrente
    - cláusula opcional *select* indica os nodos a processar
  - **<xsl:for-each select="...">**
    - processa todos os nodos indicados na cláusula *select*
  - **<xsl:sort [select="..."]>**
    - ordena os nodos (*select* indica por qual nodo ordenar)
  - **<xsl:value-of select="...">**
    - insere o valor de um nodo (especificado no *select*) na posição corrente

Exemplos de Regras XSL

- Processamento recursivo

```
<xsl:template match = "Livros">
 <html>
  <head><title>Lista de livros</title></head>
  <xsl:apply-templates/>
 </html>
    l:template>
```

formato de saida (HTML)

processa nodos filhos

- Processamento baseado em seleção

```
<xsl:template match = "livro">
 <xsl:apply-templates select = "livro[preco>100]"
</xsl:template>
```

Exemplos de Regras XSL

- Ordenação

```
<xsl:template match = “livro">
  <xsl:apply-templates>
     <xsl:sort = “titulo”>
  </xsl:apply-templates>
</xsl:template>
```

Exemplo de Transformação XSL

Entrada: doc XML

```
<listaLivros>
<livro tipo="tecnico" ISBN="01">
    <título>XML Companion<\título>
    <autor>
      <nome>N. Bradley<\nome> ...
    <\autor> ...
<\livro>
<livro tipo="tecnico" ISBN="02">
    <título>Data on the Web<\título>
    <autor>
      <nome>S. Abiteboul<\nome>...
    <\autor> ...
<\livro> ...
</listaLivros>
```

Transformação: doc XSL

```
<stylesheet xmlns = ...>
 <template match = "listaLivros">
  <html><head>
  <title>Livros Técnicos</title> </head>
    <apply-templates/>
  </html>
 </template>
 <template match = "livro">
  <P>
   <apply-templates select =
        "livro[@tipo = "tecnico"]">
      <sort = "título">
    </apply-templates>
  </P>
 </template>
 ...
```

processar elementos filhos

selecionar livros técnicos

ordenar por título

Exemplo de Transformação XSL

Entrada: doc XML

```
<listaLivros>
<livro tipo="tecnico" ISBN="01">
    <título>XML Companion<\título>
    <autor>
      <nome>N. Bradley<\nome> ...
    <\autor> ...
<\livro>
<livro tipo="tecnico" ISBN="02">
    <título>Data on the Web<\título>
    <autor>
      <nome>S. Abiteboul<\nome>...
    <\autor> ...
<\livro> ...
</listaLivros>
```

Transformação: doc XSL

```
...
<variable name =
"separador">,</variable>

 <template match = "título">
    <value-of select = ".">
    <value-of select = "{$separador}">

 </template>

  <template match = "autor/nome">
     <value-of select = ".">
  </template>
</stylesheet>
```

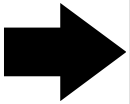
selecionar o conteúdo de título

selecionar o conteúdo do nome do autor

Exemplo de Transformação XSL

Entrada: doc XML

```
<listaLivros>
<livro tipo="tecnico" ISBN="01">
    <título>XML Companion<\título>
    <autor>
      <nome>N. Bradley<\nome> ...
    <\autor> ...
<\livro>
<livro tipo="tecnico" ISBN="02">
    <título>Data on the Web<\título>
    <autor>
      <nome>S. Abiteboul<\nome>...
    <\autor> ...
<\livro> ...
</listaLivros>
```



Saída: doc HTML

```
<html>
   <head>
     <title>
          Livros Técnicos
      </title>
   </head>
  <P>
    XML Companion,N. Bradley
  </P>
  <P>
    Data on the Web,S. Abiteboul
  </P>
...
</html>
```