XPath

- Primeira recomendação para consulta a dados
- Linguagem para o acesso a partes de um doc XML
  - sintaxe: expressões de caminho
    - assemelha-se à navegação em diretórios de arquivos
  - exemplo
    - expressão *XPath*: `/listaLivros/livro/título`
    - resultado:

```
<título>Tecnologia XML</título>
<título>Sistema de Banco de Dados</título>
...
```

XPath - Exemplos

`/listaLivros` (elemento raiz – todo o doc XML)

`/listaLivros/livro/*/eMail` ('*' substitui 1 elem)

`/listaLivros/livro//seção` (qq elemento descendente seção)

`/listaLivros/livro/capítulo[1]` (primeiro capítulo de livros)

`/listaLivros/livro/capítulo/nome |`

`/listaLivros/livro/capítulo/seção/nome` (união)

`/listaLivros/livro/@ISBN` (acesso a um atributo)

`/listaLivros/livro[título = "XML"]` (filtro)

`/listaLivros/livro[capitulo/@nome = "XML" or //seção/nome = "XML"]/título` (filtro)

`/listaLivros/livro//seção[last()]` (função)

XQuery

- Recomendação mais recente
- Recursos adicionais em relação à *XPath*
  - junções, definição de estruturas de resultado, variáveis de consulta, atributos calculados, funções de agregação, ...
- Sintaxe básica (expressão "FLWR")

```
for variável in expressãoXPath
[let associação de novas variáveis]
[where condição]
return estrutura de resultado
```

XQuery - Exemplos

```
for $liv in /listaLivros/livro
where $liv/autor/nome = "João Silva"
return { $liv/@ISBN, $liv/titulo }
```

(consulta simples)

```
for $liv in /listaLivros/livro
let $pDesc := $liv/preço - $liv/preço * 0.1
where $liv/categoria = "ficcao"
return <FiccaoDesc>{$liv/titulo, $pDesc}</FiccaoDesc>
```

```
for $liv1 in /listaLivros/livro[@ISBN = "562"]
for $liv2 in /listaLivros/livro
where $liv2/@ISBN != $liv1/@ISBN
and $liv2/autor/nome = $liv1/autor/nome
return $liv2/titulo
```

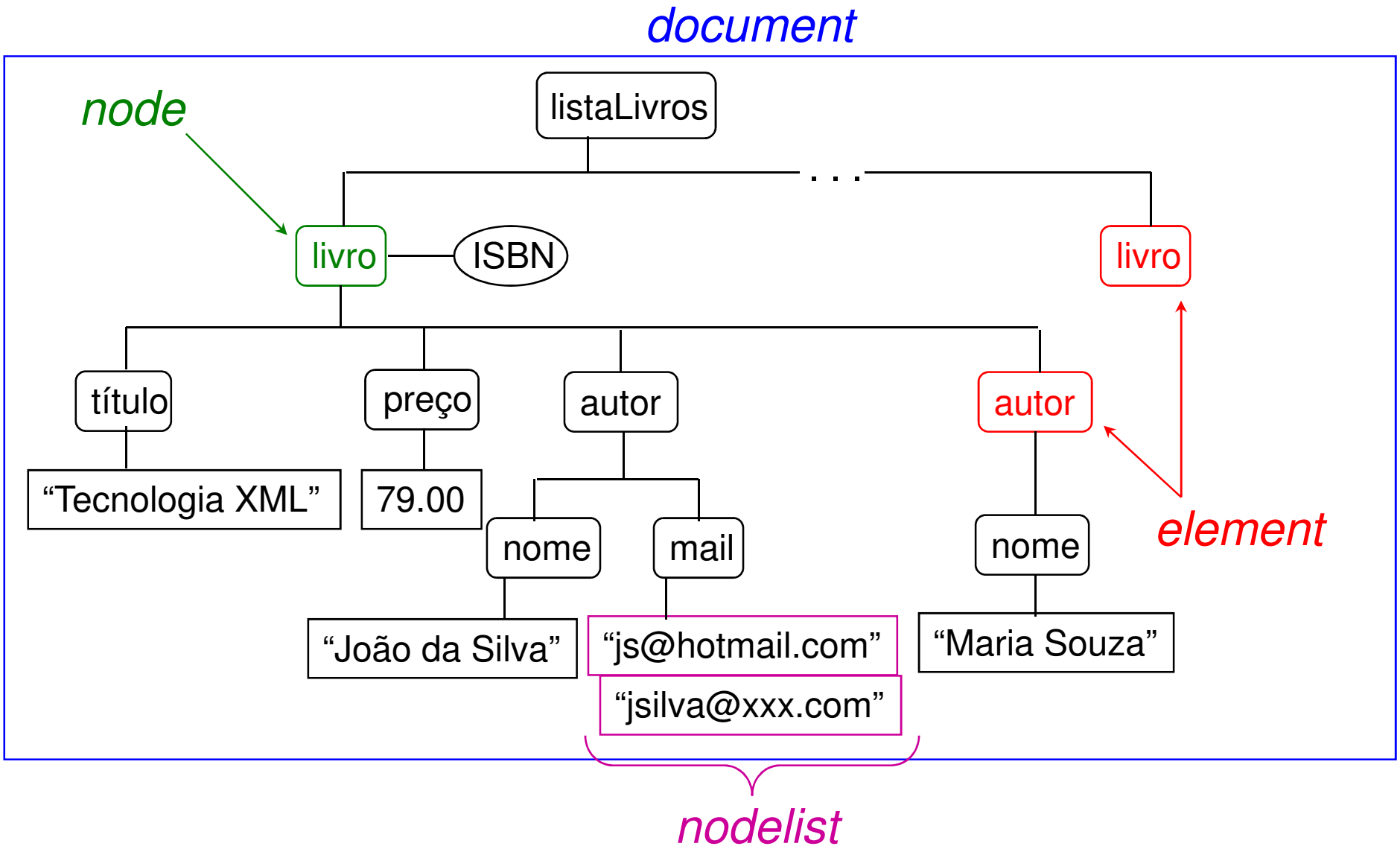
(nova estrutura de resultado)

(junção)

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*

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

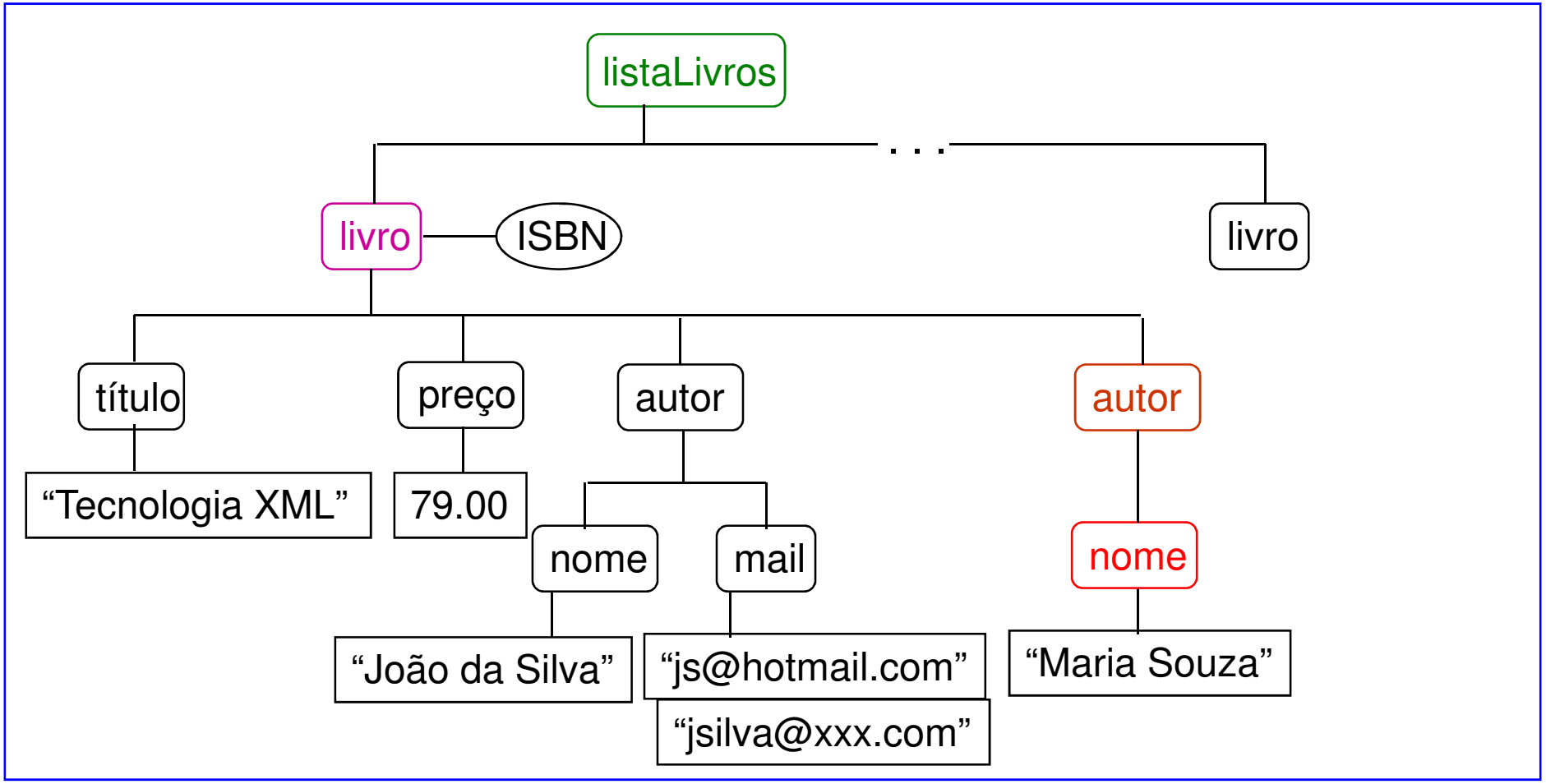
element

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)
  - define regras para a apresentação de dados
- XSL
  - linguagem de definição de folha de estilos para um doc XML
    - formatação de apresentação
    - 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

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>
```

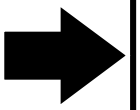
selecionar o conteúdo de título

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>
```