University of Jijel
Faculty of Exact Sciences and Computer Science
Department of Computer Science
L3 – Computer Systems

Semi-Structured Data
Chapter 4
XML Galaxy

Tarek Boutefara
t_boutefara@univ-jijel.dz

2025/2026

Content

- DOM
- SAX
- XPATH
- XSL
- Example : SVG
- XLINK
- XPOINTER

Content

- **DOM**
- SAX
- XPATH
- XSL
- Example : SVG
- XLINK
- XPOINTER

DOM

- Definition
  - An abstract programming interface that allows an XML (or HTML) document to be represented as a tree of nodes.

DOM

- Definition
  - Everything is a node
    - Elements,
    - Attributes,
    - DocumentType,
    - Comment,
    - Text,
    - ...

DOM

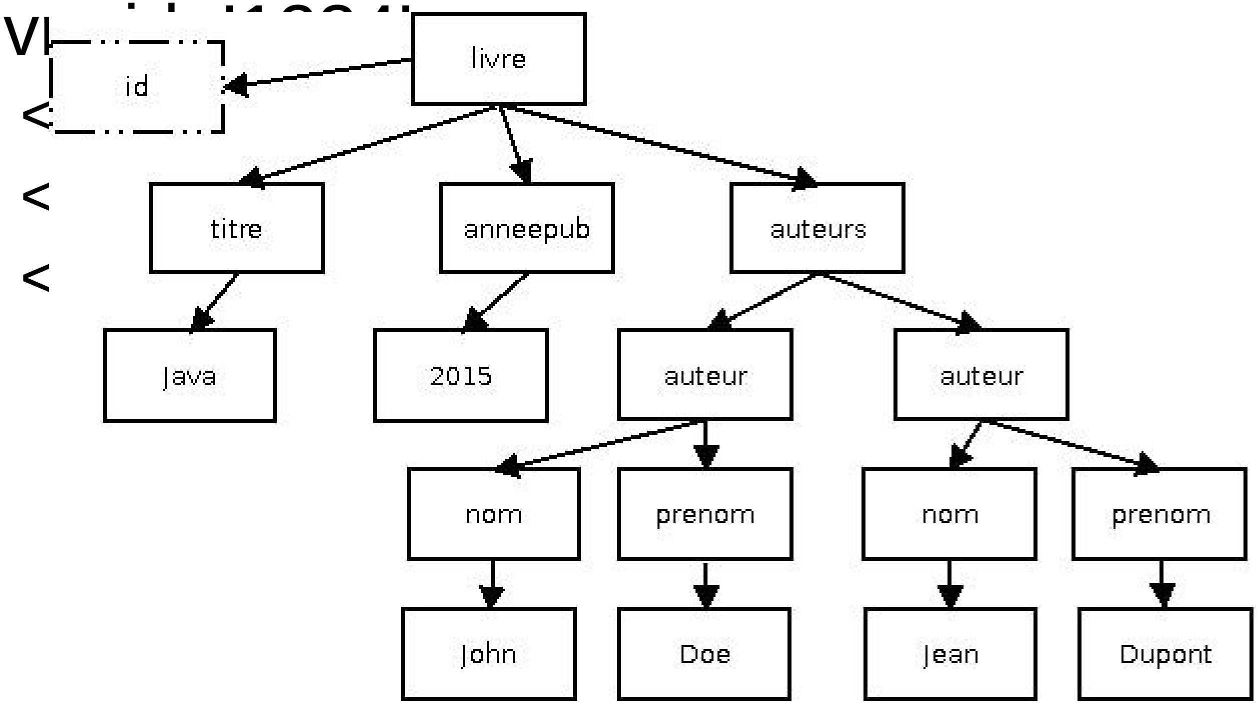
- Navigation:
  - A document represents the root of the DOM.
  - You can retrieve the children of an element.
  - You can access its neighbors (siblings).
  - You can retrieve the first child.
  - You can retrieve the last child.
  - ...

DOM

- <livre id='1234'>
  - <titre>Java</titre>
  - <anneepub>2015</anneepub>
  - <auteurs>
    - <auteur>
      - <nom>John</nom><prenom>Doe</prenom>
    - </auteur>
    - <auteur>
      - <nom>John</nom><prenom>Doe</prenom>
    - </auteur>
  - </auteurs>
- </livre>

DOM

- <liv
  - <
  - <
  - <


livre
id
titre
anneepub
auteurs
Java
2015
auteur
auteur
nom
prenom
nom
prenom
John
Doe
Jean
Dupont

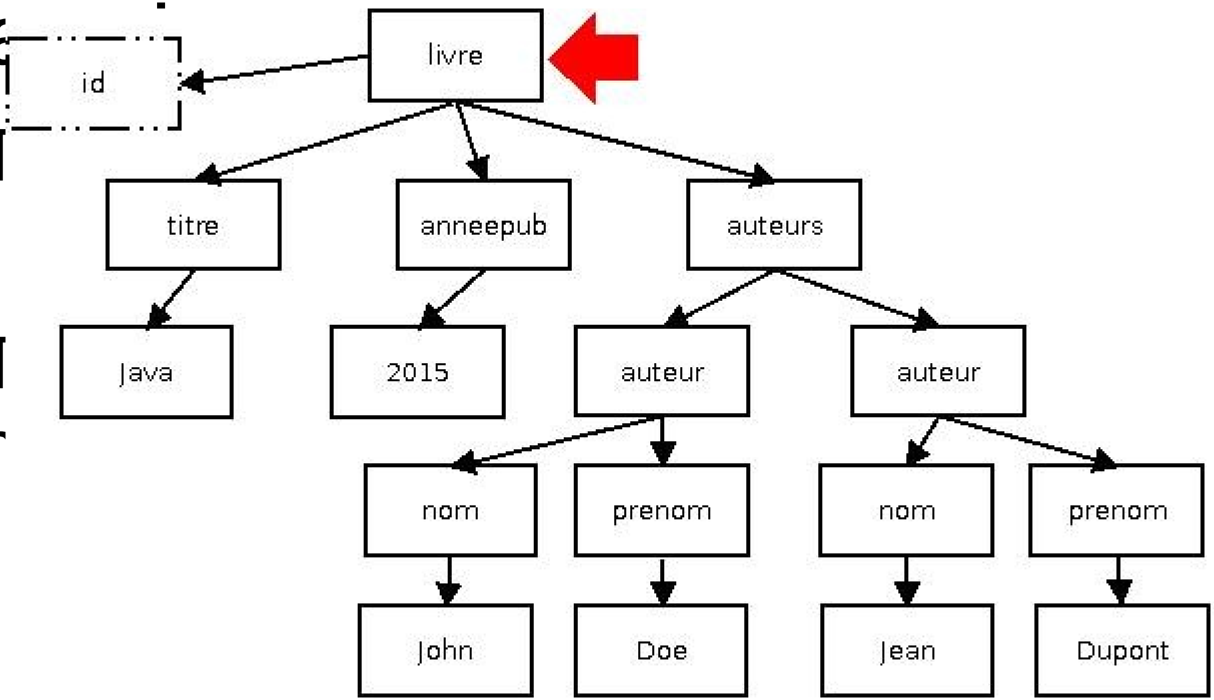
  - </auteurs>
- </livre>

DOM

- Example:
  - To retrieve the root in JS:
    - xmlDoc.documentElement
  - This method returns an "element" (that is, a node)

DOM

- Exa
  - T
  - T at is, a r

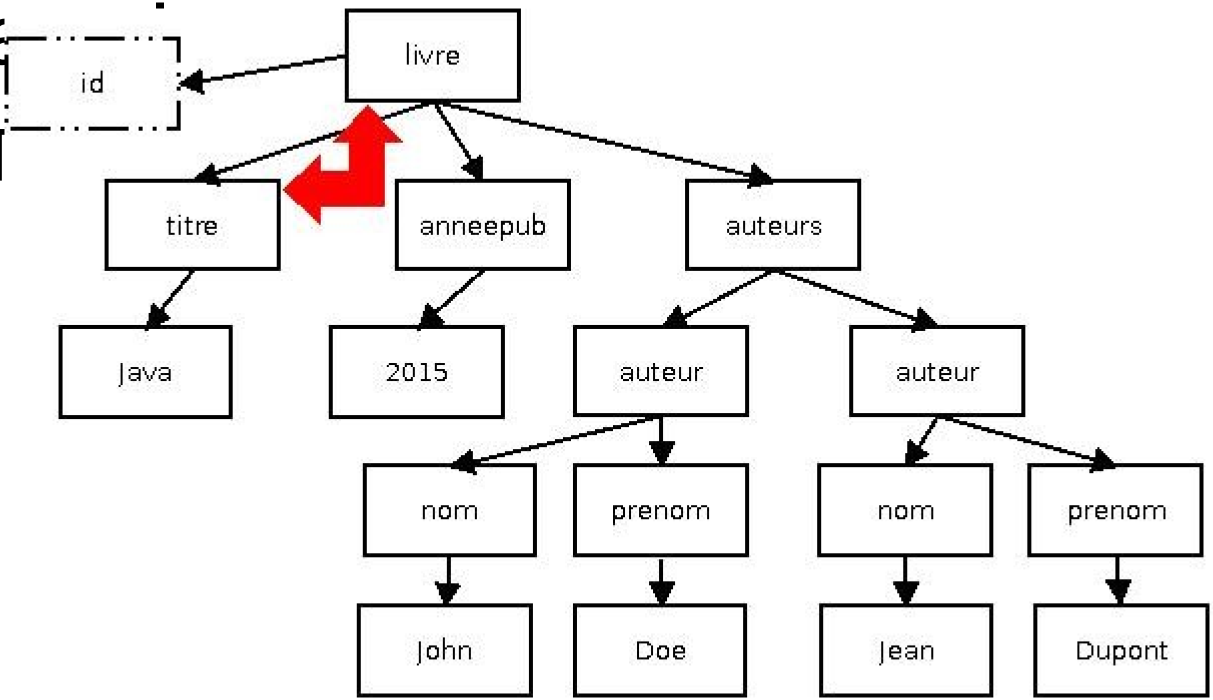


DOM

- Example:
  - To retrieve the first child in Javascript:
    - element.firstElementChild

DOM

- Exa
  - T

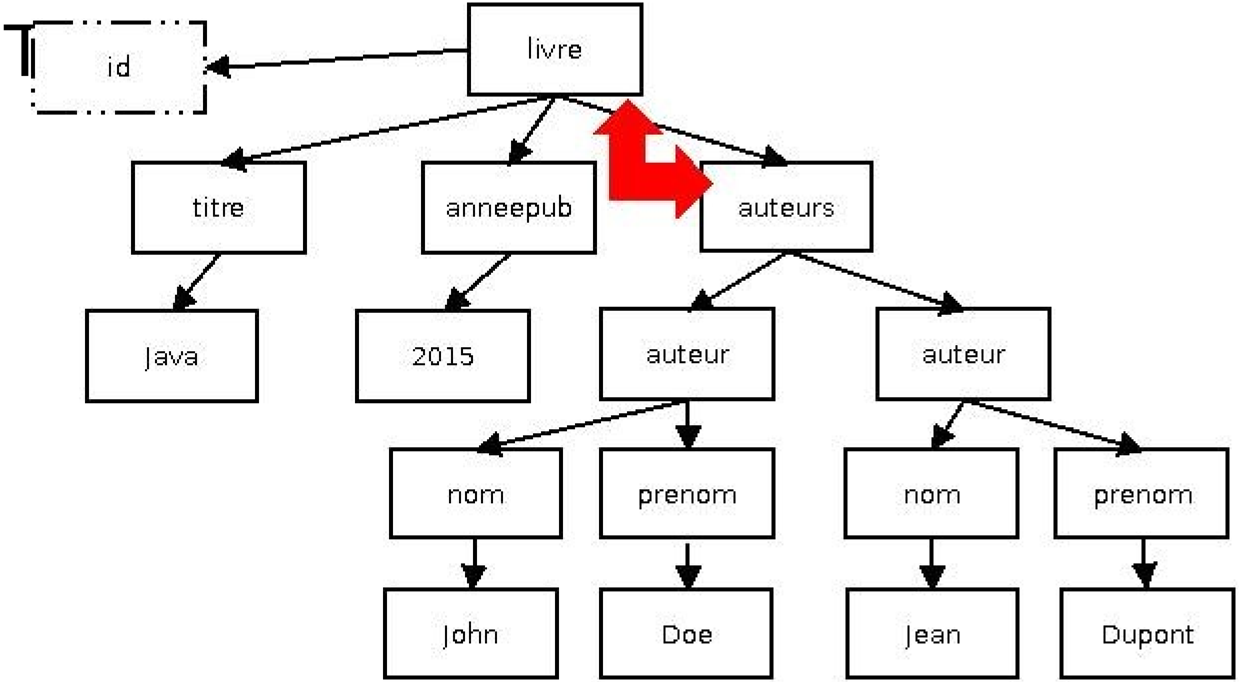

livre
id
titre
anneepub
auteurs
Java
2015
auteur
auteur
nom
prenom
nom
prenom
John
Doe
Jean
Dupont

DOM

- Example:
  - To retrieve the last child in Javascript:
    - element.lastElementChild

DOM

- Example:
  - T

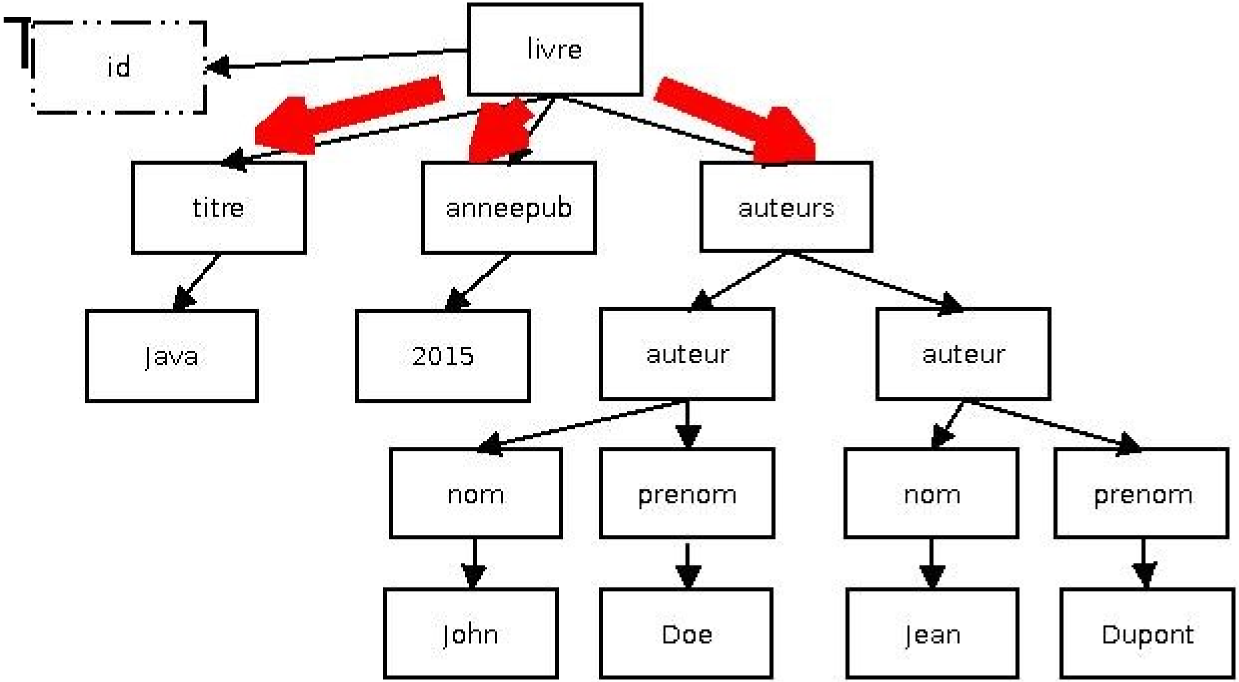


DOM

- Example:
  - To retrieve all children in Javascript:
    - element.children
      - As an HTMLCollection
    - element.childNodes
      - As a NodeList

DOM

- Example:
  - T

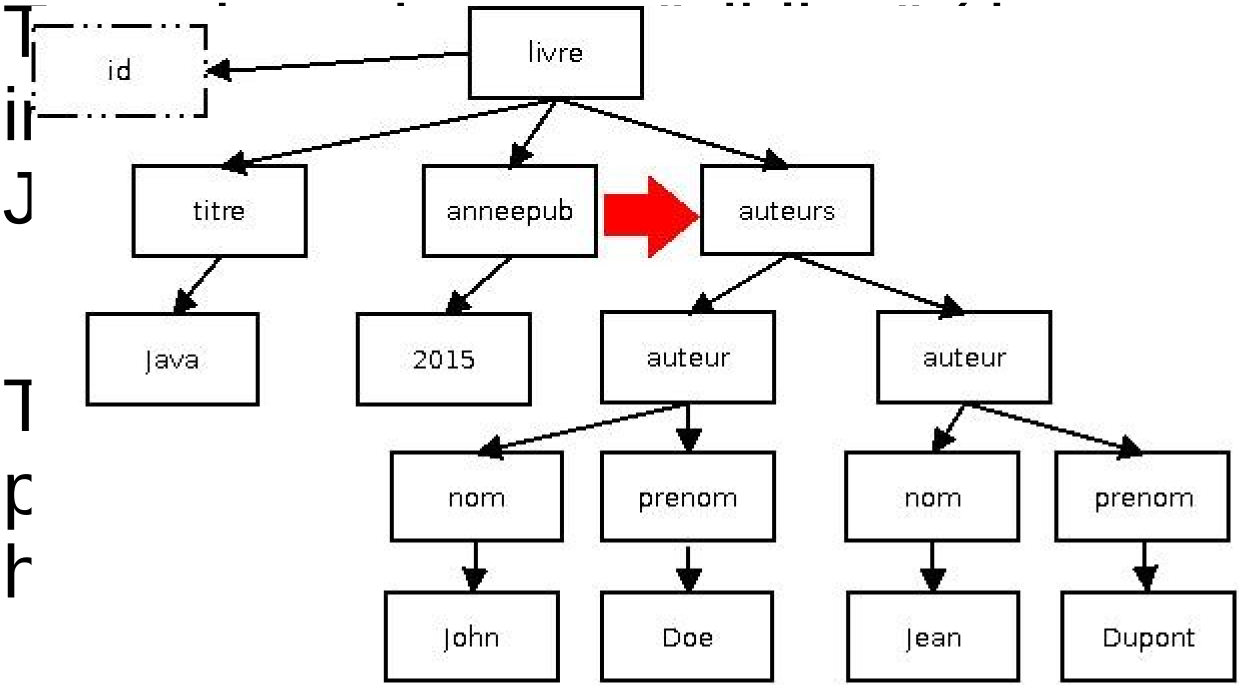


DOM

- Example:
  - To retrieve the next "sibling" (the next node in the same level of the hierarchy) in Javascript:
    - element.nextElementSibling
  - To retrieve the previous "sibling" (the previous node in the same level of the hierarchy) in Javascript:
    - element.previousElementSibling

DOM

- Example:
  - T node
    i
    J
  - T
    p e
    h
    elementpreviousElementSibling

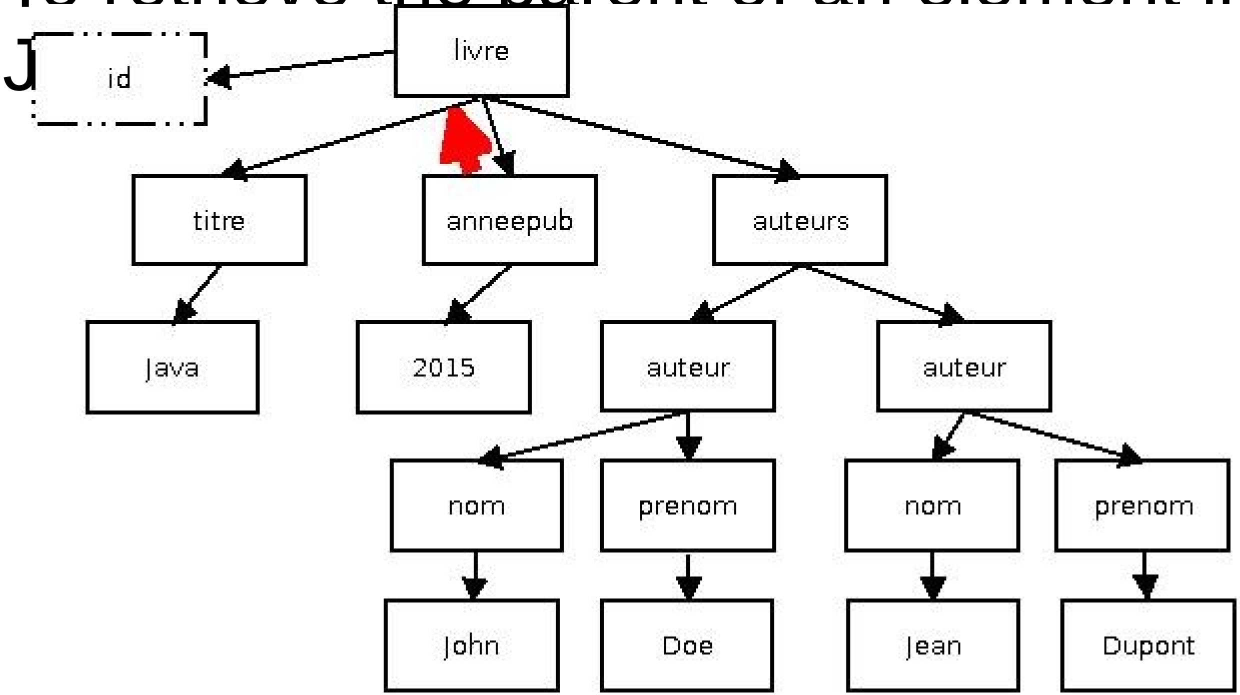


DOM

- Example:
  - To retrieve the parent of an element in JavaScript:
    - element.parentElement

DOM

- Example:
  - To retrieve the parent of an element in J



DOM

- Example:
  - It is also possible to retrieve all nodes and not just the elements (tags),
    - element.firstChild
    - element.lastChild
    - ...

DOM

- Example:
  - It is also possible to retrieve the element directly using:
    - The ID
      - xmlDoc.getElementById()
    - The tag name
      - xmlDoc.getElementsByTagName()
    - The name attribute
      - xmlDoc.getElementsByName()

DOM

- Important notes:
  - When working with HTML, we already have the "document" object and no parsing is necessary.

DOM

- Important notes:
  - When working with XML, you must first create the document by parsing the XML content:
    - var text = "<book><title>Java</title>" +
    - "<year>2005</year>" +
    - "<price>19.99</price></book>";
    - var parser = new DOMParser();
    - var xmlDoc = parser.parseFromString(text,"text/xml");

Content

- DOM
- **SAX**
- XPATH
- XSL
- Exemple : SVG
- XLINK
- XPOINTER

SAX

- SAX
  - Simple API for XML,
  - Allows you to parse an XML file without creating the DOM tree,
  - Based on the principle of events:
    - It reads the file from beginning to end,
    - Each time, an XML element is encountered, an event is triggered.

SAX

- Development method under SAX:
  - Node manipulation is used.
  - Methods are written to receive and process events.
    - This is very similar to the Listener principle used by Java Swing.
    - Each method is specialized (called) for a given type of event.
    - Upon reception, the event (the XML elements) is passed to the appropriate method.

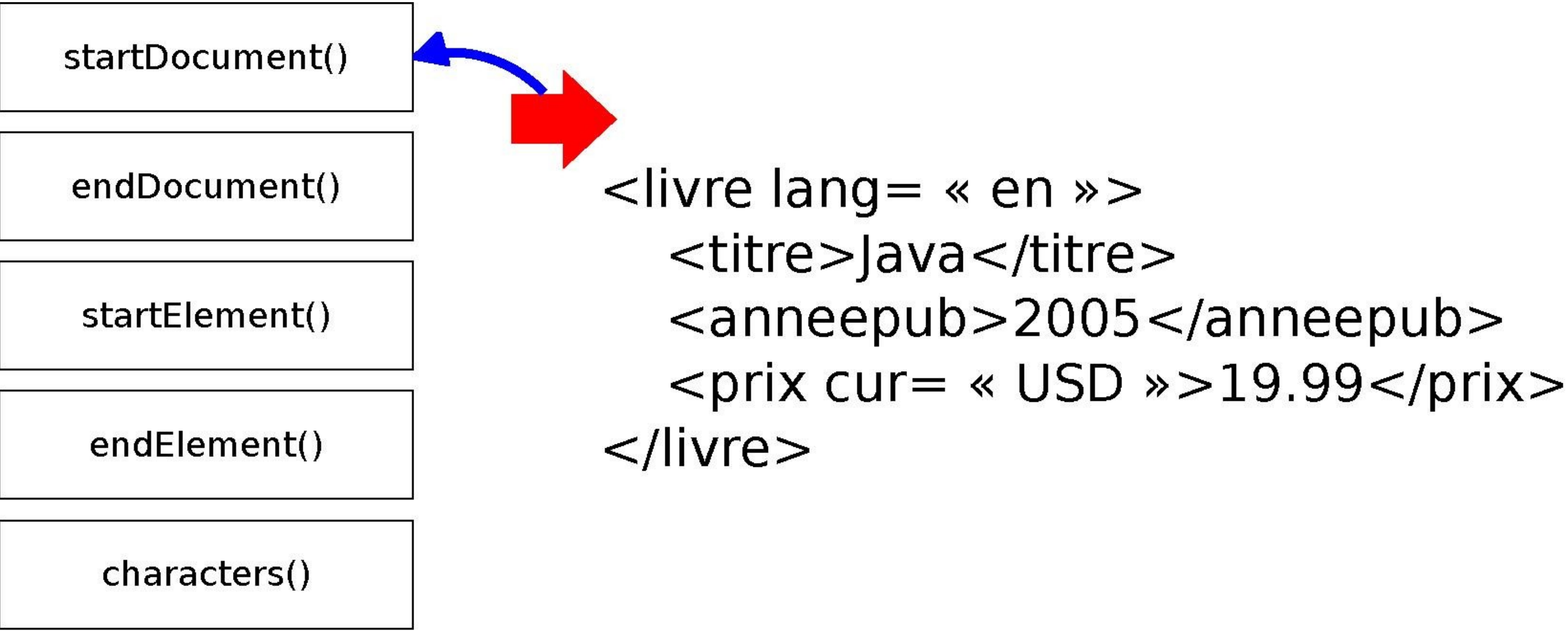
SAX

- Development method under SAX:
  - List of events:
    - Varies depending on the platform and parser used.
    - Under Java, the class to extend is the DefaultHandler class:
    - https://docs.oracle.com/javase/7/docs/api/org/xml/sax/helpers/DefaultHandler.html

SAX

- Development method under SAX, example:
  - <book lang="en">
    - <title>Java</title>
    - <year>2005</year>
    - <price cur="USD">19.99</price>
  - </book>

SAX



startDocument()

endDocument()

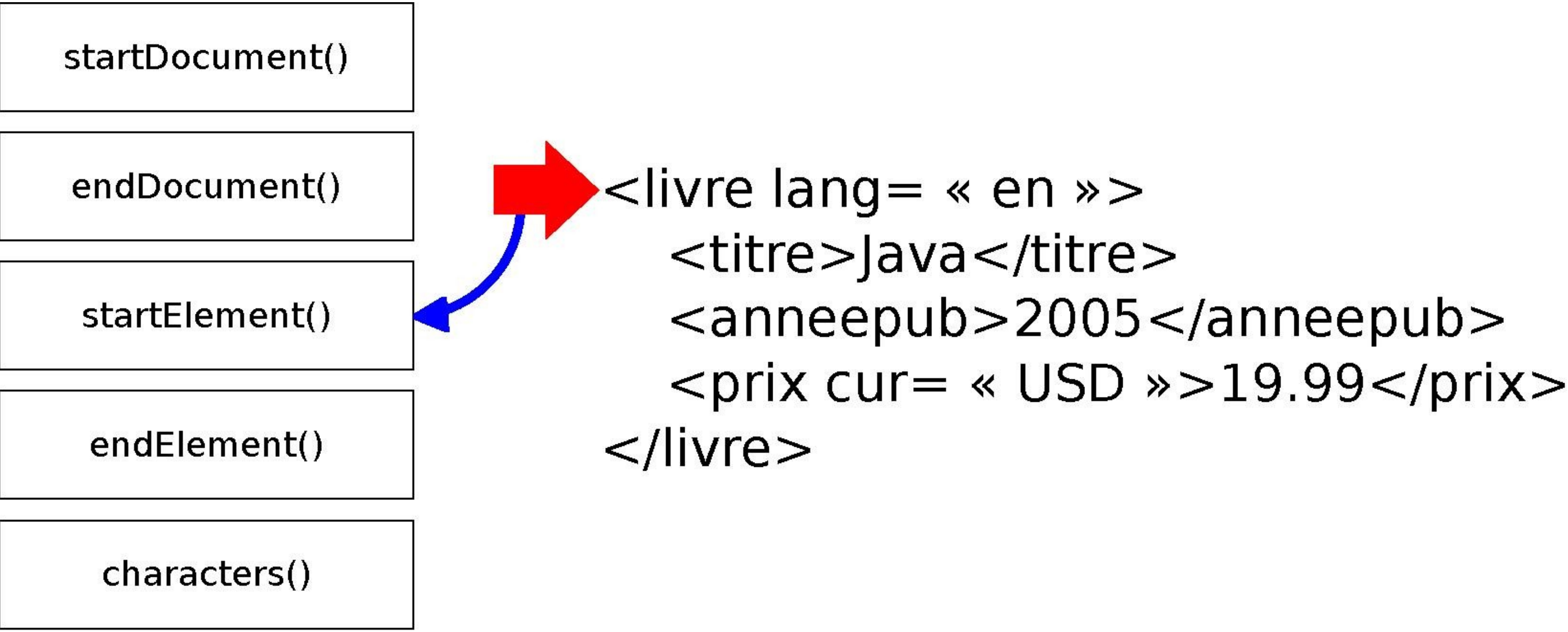
startElement()

endElement()

characters()

```
<livre lang= « en »>
    <titre>Java</titre>
    <anneepub>2005</anneepub>
    <prix cur= « USD »>19.99</prix>
</livre>
```

SAX



startDocument()

endDocument()

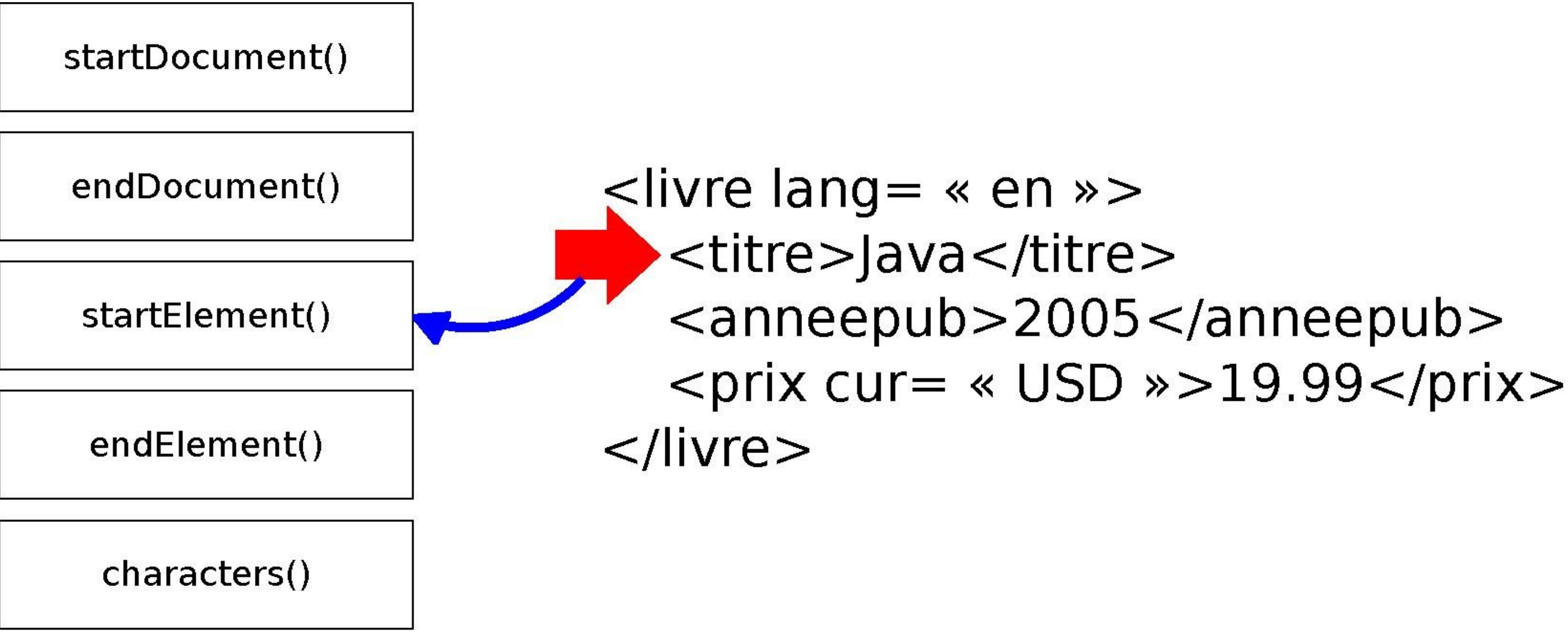
startElement()

endElement()

characters()

```
<livre lang= « en »>
    <titre>Java</titre>
    <anneepub>2005</anneepub>
    <prix cur= « USD »>19.99</prix>
</livre>
```

SAX



startDocument()

endDocument()

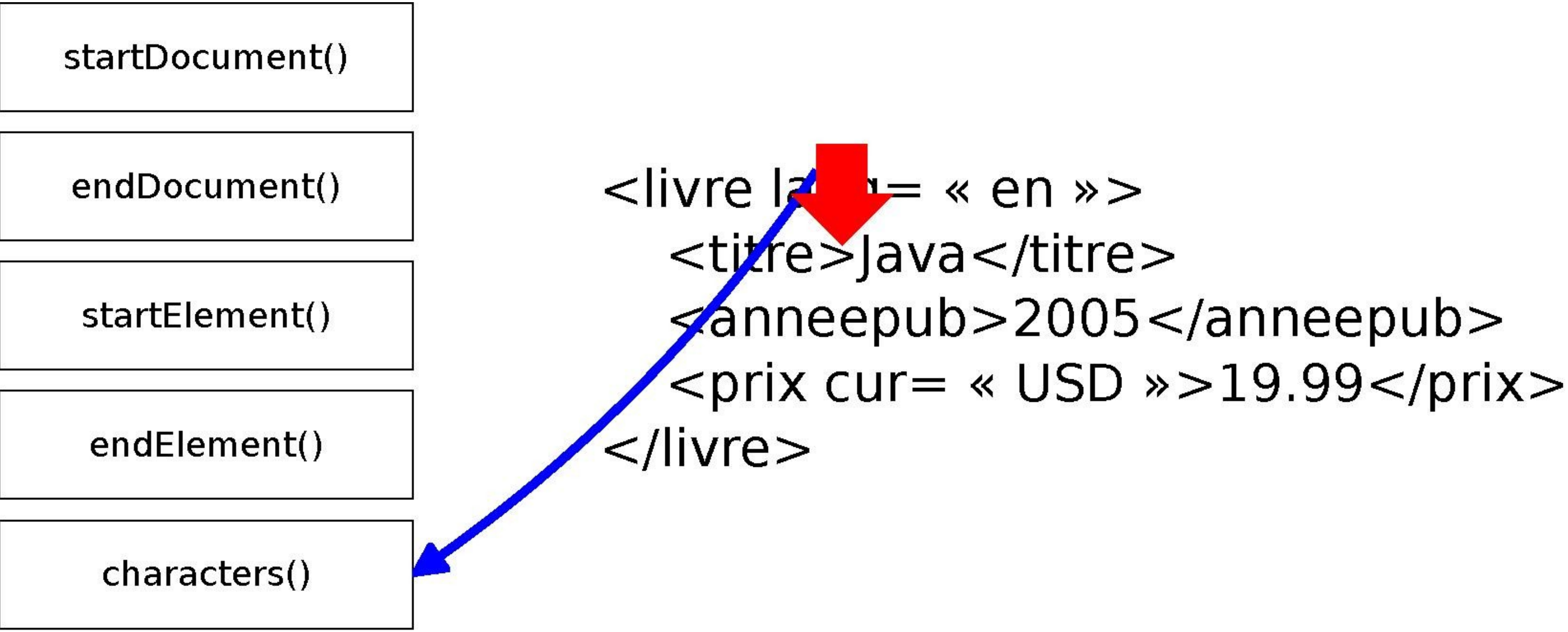
startElement()

endElement()

characters()

```
<livre lang= « en »>
    <titre>Java</titre>
    <anneepub>2005</anneepub>
    <prix cur= « USD »>19.99</prix>
</livre>
```

SAX



startDocument()

endDocument()

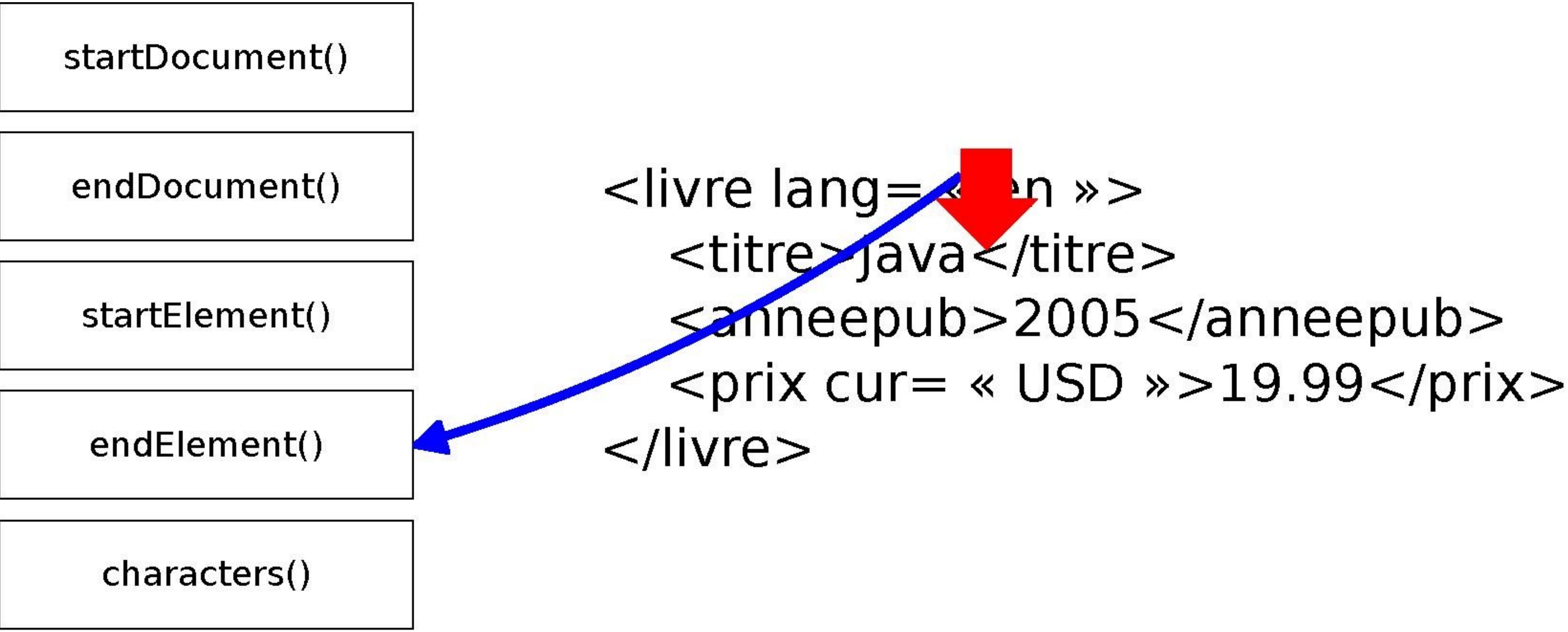
startElement()

endElement()

characters()

```
<livre lang= « en »>
    <titre>Java</titre>
    <anneepub>2005</anneepub>
    <prix cur= « USD »>19.99</prix>
</livre>
```

SAX



startDocument()

endDocument()

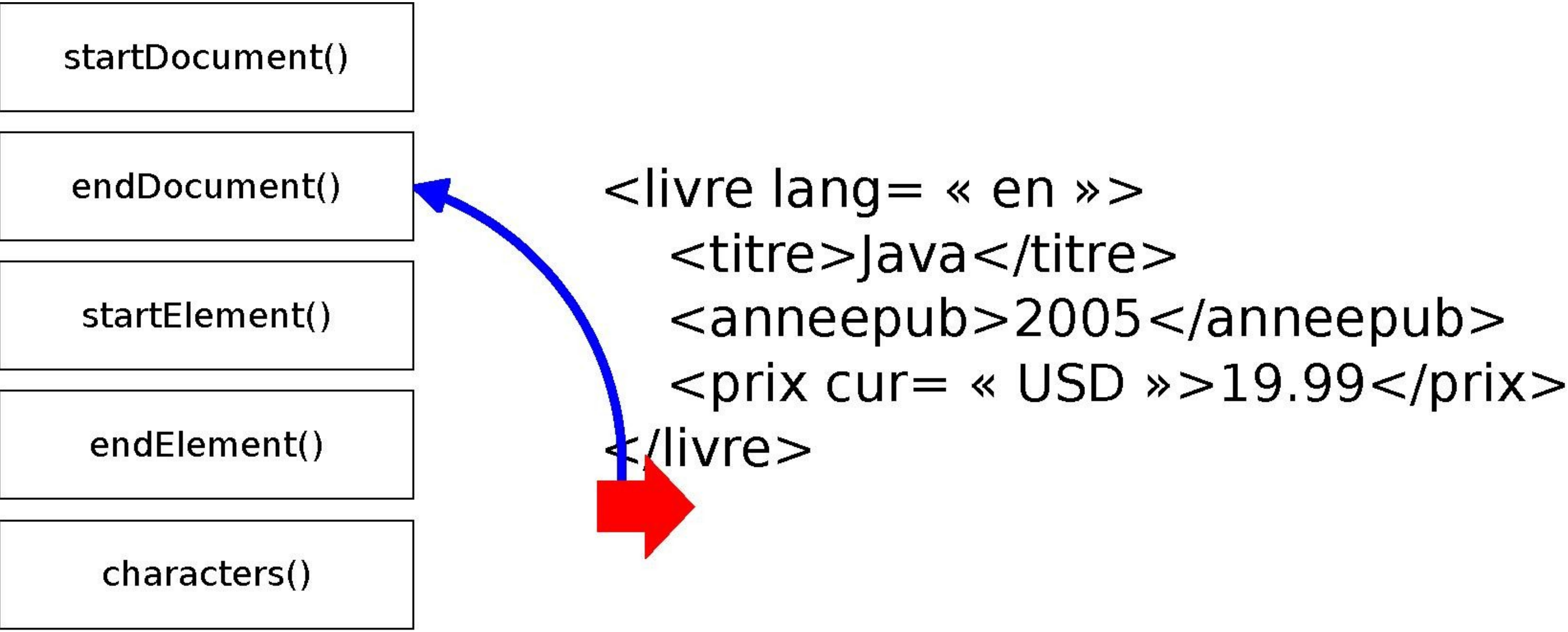
startElement()

endElement()

characters()

```
<livre lang= « en »>
    <titre>Java</titre>
    <anneepub>2005</anneepub>
    <prix cur= « USD »>19.99</prix>
</livre>
```

SAX



startDocument()

endDocument()

startElement()

endElement()

characters()

```
<livre lang= « en »>
    <titre>Java</titre>
    <anneepub>2005</anneepub>
    <prix cur= « USD »>19.99</prix>
</livre>
```

SAX

- SAX vs. DOM
  - SAX consumes less space and is faster.
  - However,
    - it does not allow navigation within the structure like DOM,
    - and it is inefficient for creating and querying.
  - SAX is used to process large files with a small depth.

Content

- DOM
- SAX
- **XPATH**
- XSL
- Exemple : SVG
- XLINK
- XPOINTER

XPath

- Definition
  - XPath is an abstract language for querying XML documents.
  - This query returns a subset of the nodes (or part of the document).
  - It was proposed by the W3C.

XPath

- Principle
  - XPath constructs the logical tree of the XML document.
  - The passed expression is evaluated against this tree.
  - The passed expression is in the form of a "path"
    - inspired by the file paths used on the hard drive.

XPath

- Principle
  - The expression can point to all types of nodes:
    - Elements,
    - Attributes,
    - Entities,
    - Etc.

XPath

- Principle
  - Evaluation is done in the order of appearance in the document.
    - The order of the nodes is important,
    - unlike the relational model of databases where the order is not important.

XPath

- Expressions:
  - Node:
  - <example>Hello</example>
    - Element,
  - Lang = "en"
    - Node attribute
  - Atomic value (of nodes without parent and without children):
    - Hello
    - "en"

XPath

- The expressions:
  - Relationships:
    - Parent: each element or attribute has only one parent,
    - Child: each element can have one or more children or it can have none,
    - Brothers: nodes can have the same parent,
    - Ancestors: the parent, the parent of the parent, etc.,
    - Descendants: the children, the children of the children, etc.

XPath

- Expressions:
    - /step/step/step/...|/step/step/...|...

XPath

- The expressions:
  - `node_name`: select nodes with the name "node_name",
  - `//`: select from the current node regardless of their position,
  - `/`: select from the root,
  - `.`: select the current node,
  - `..`: select the parent node,
  - `@`: select attribute.

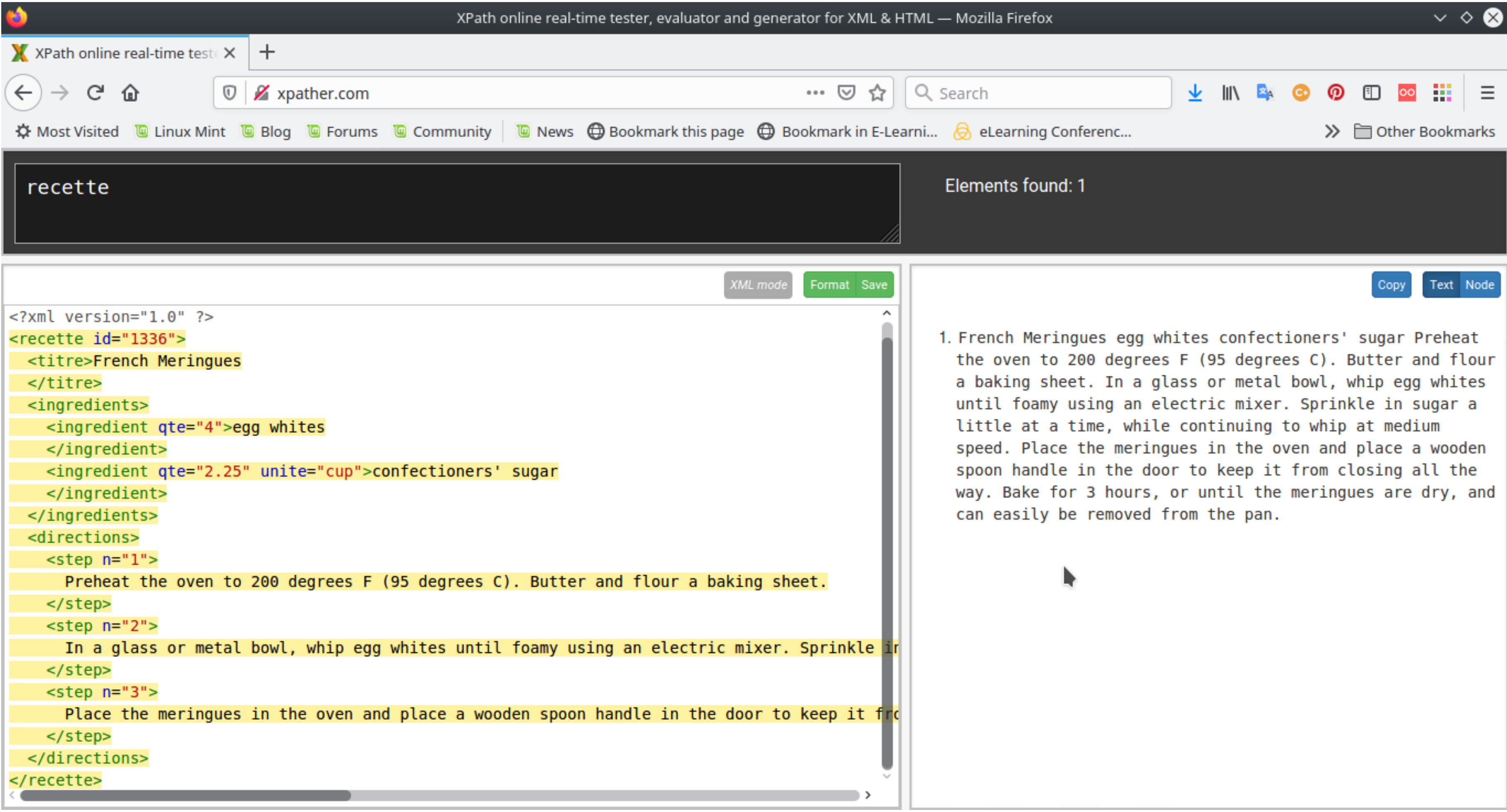
XPath

- Example
  - Recipe.xml

XPath

- Example
  - Recipe.xml
    - Select all :
      - recipe

XPath



```
recette
```

Elements found: 1

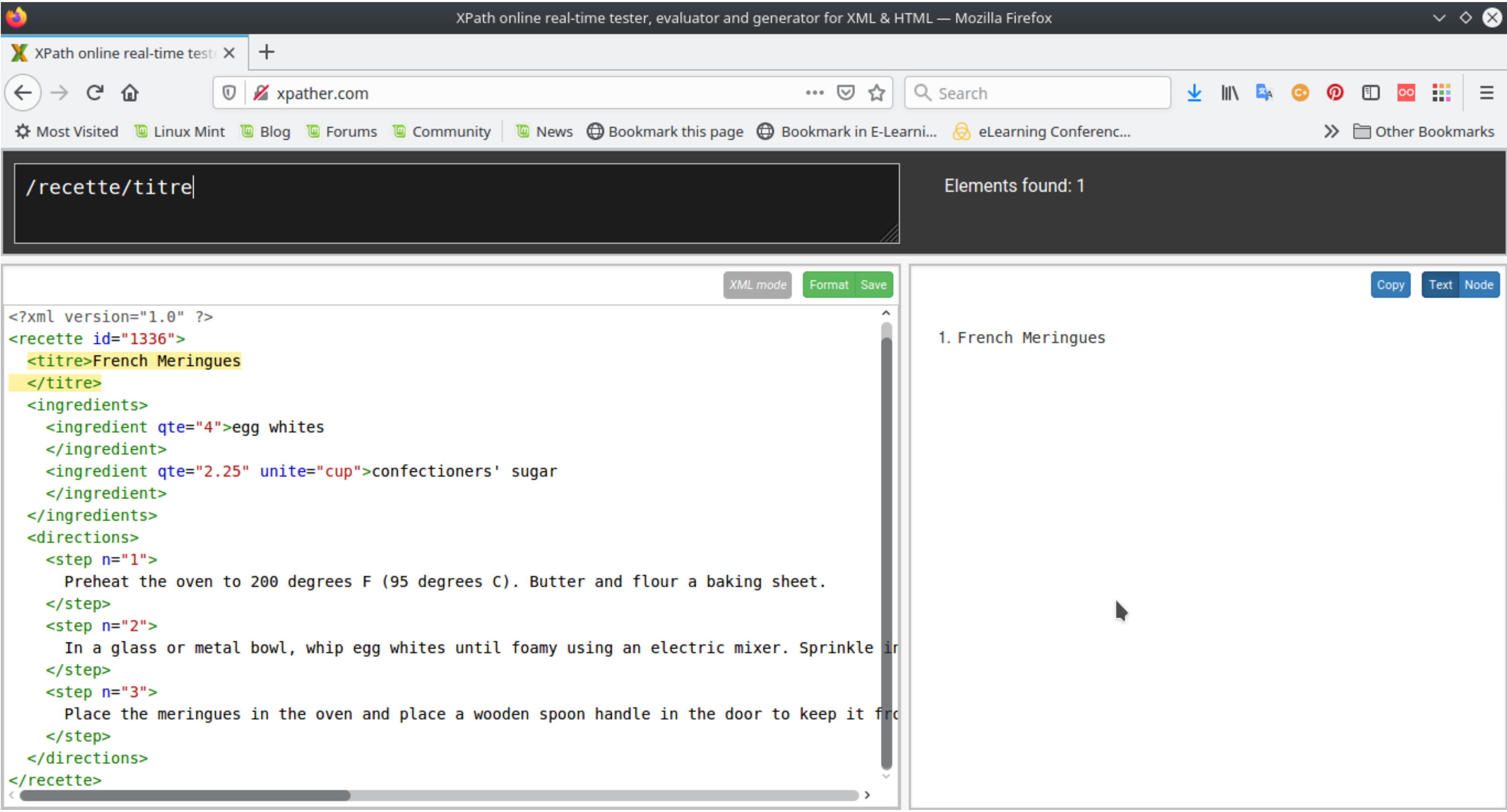
```
<?xml version="1.0" ?>
<recette id="1336">
  <titre>French Meringues
  </titre>
  <ingredients>
    <ingredient qte="4">egg whites
    </ingredient>
    <ingredient qte="2.25" unite="cup">confectioners' sugar
    </ingredient>
  </ingredients>
  <directions>
    <step n="1">
      Preheat the oven to 200 degrees F (95 degrees C). Butter and flour a baking sheet.
    </step>
    <step n="2">
      In a glass or metal bowl, whip egg whites until foamy using an electric mixer. Sprinkle ir
    </step>
    <step n="3">
      Place the meringues in the oven and place a wooden spoon handle in the door to keep it frc
    </step>
  </directions>
</recette>
```

```
1. French Meringues egg whites confectioners' sugar Preheat
   the oven to 200 degrees F (95 degrees C). Butter and flour
   a baking sheet. In a glass or metal bowl, whip egg whites
   until foamy using an electric mixer. Sprinkle in sugar a
   little at a time, while continuing to whip at medium
   speed. Place the meringues in the oven and place a wooden
   spoon handle in the door to keep it from closing all the
   way. Bake for 3 hours, or until the meringues are dry, and
   can easily be removed from the pan.
```

XPath

- Example
  - Recipe.xml
    - Select the title :
      - /recipe/title

XPath



```
/recette/titre
```

Elements found: 1

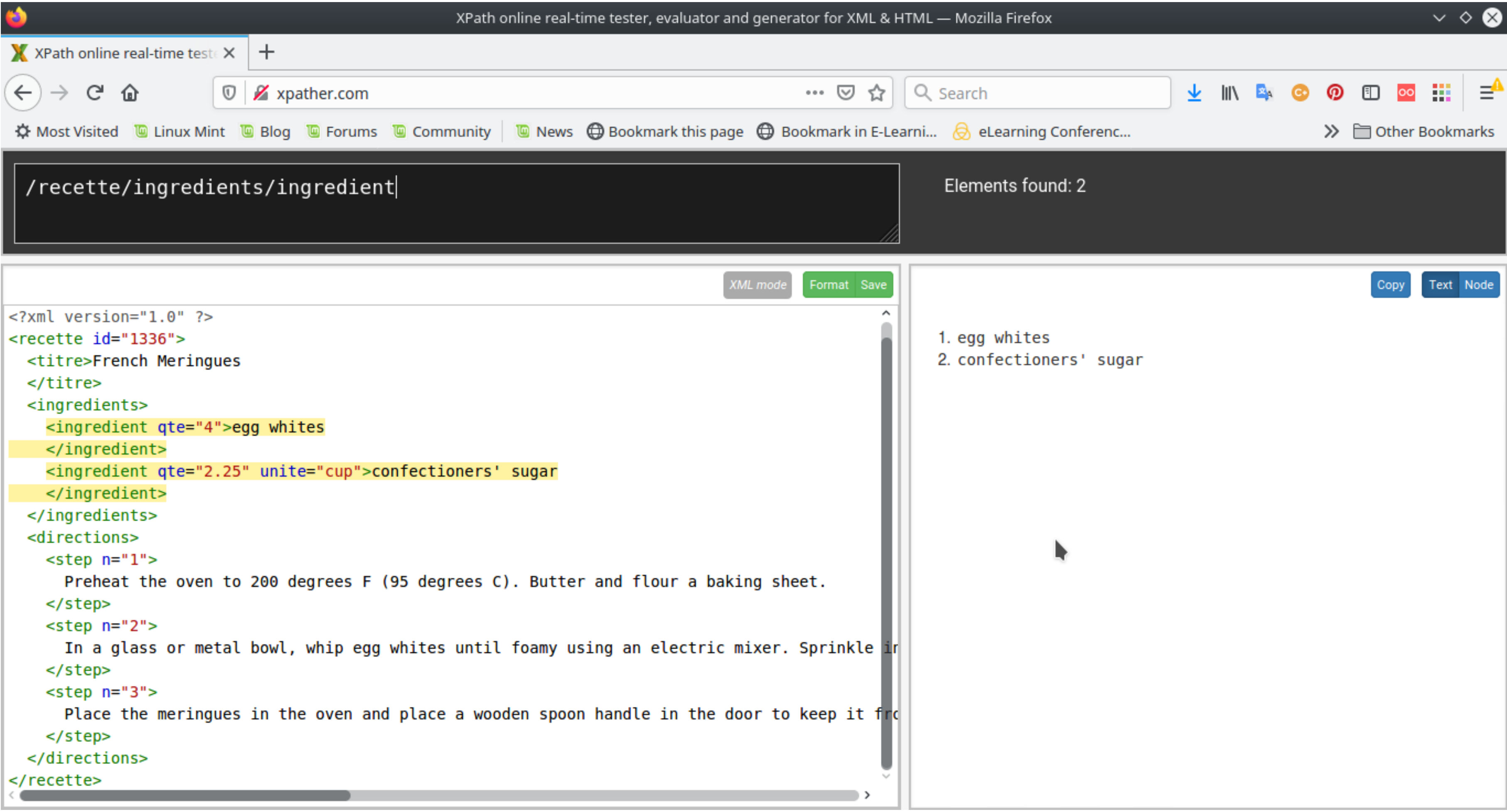
```
<?xml version="1.0" ?>
<recette id="1336">
  <titre>French Meringues
  </titre>
  <ingredients>
    <ingredient qte="4">egg whites
    </ingredient>
    <ingredient qte="2.25" unite="cup">confectioners' sugar
    </ingredient>
  </ingredients>
  <directions>
    <step n="1">
      Preheat the oven to 200 degrees F (95 degrees C). Butter and flour a baking sheet.
    </step>
    <step n="2">
      In a glass or metal bowl, whip egg whites until foamy using an electric mixer. Sprinkle ir
    </step>
    <step n="3">
      Place the meringues in the oven and place a wooden spoon handle in the door to keep it fro
    </step>
  </directions>
</recette>
```

```
1. French Meringues
```

XPath

- Example
  - Recipe.xml
    - Select all ingredients :
      - /recipe/ingredients/ingredient
      - /recipe//ingredient

XPath



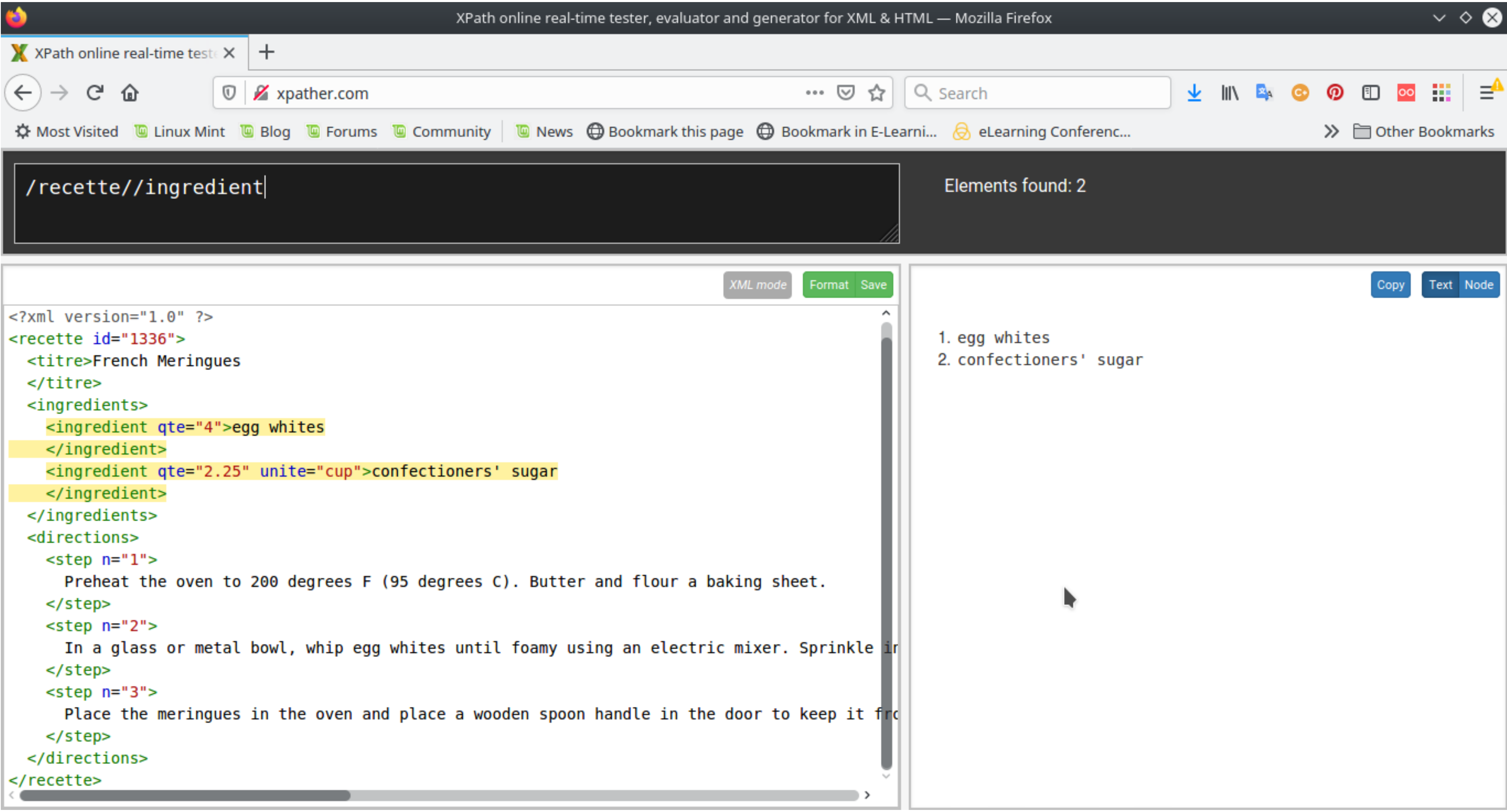
```
/recette/ingredients/ingredient
```

Elements found: 2

```
<?xml version="1.0" ?>
<recette id="1336">
  <titre>French Meringues
  </titre>
  <ingredients>
    <ingredient qte="4">egg whites
    </ingredient>
    <ingredient qte="2.25" unite="cup">confectioners' sugar
    </ingredient>
  </ingredients>
  <directions>
    <step n="1">
      Preheat the oven to 200 degrees F (95 degrees C). Butter and flour a baking sheet.
    </step>
    <step n="2">
      In a glass or metal bowl, whip egg whites until foamy using an electric mixer. Sprinkle ir
    </step>
    <step n="3">
      Place the meringues in the oven and place a wooden spoon handle in the door to keep it frc
    </step>
  </directions>
</recette>
```

```
1. egg whites
2. confectioners' sugar
```

XPath



```
/recette//ingredient
```

Elements found: 2

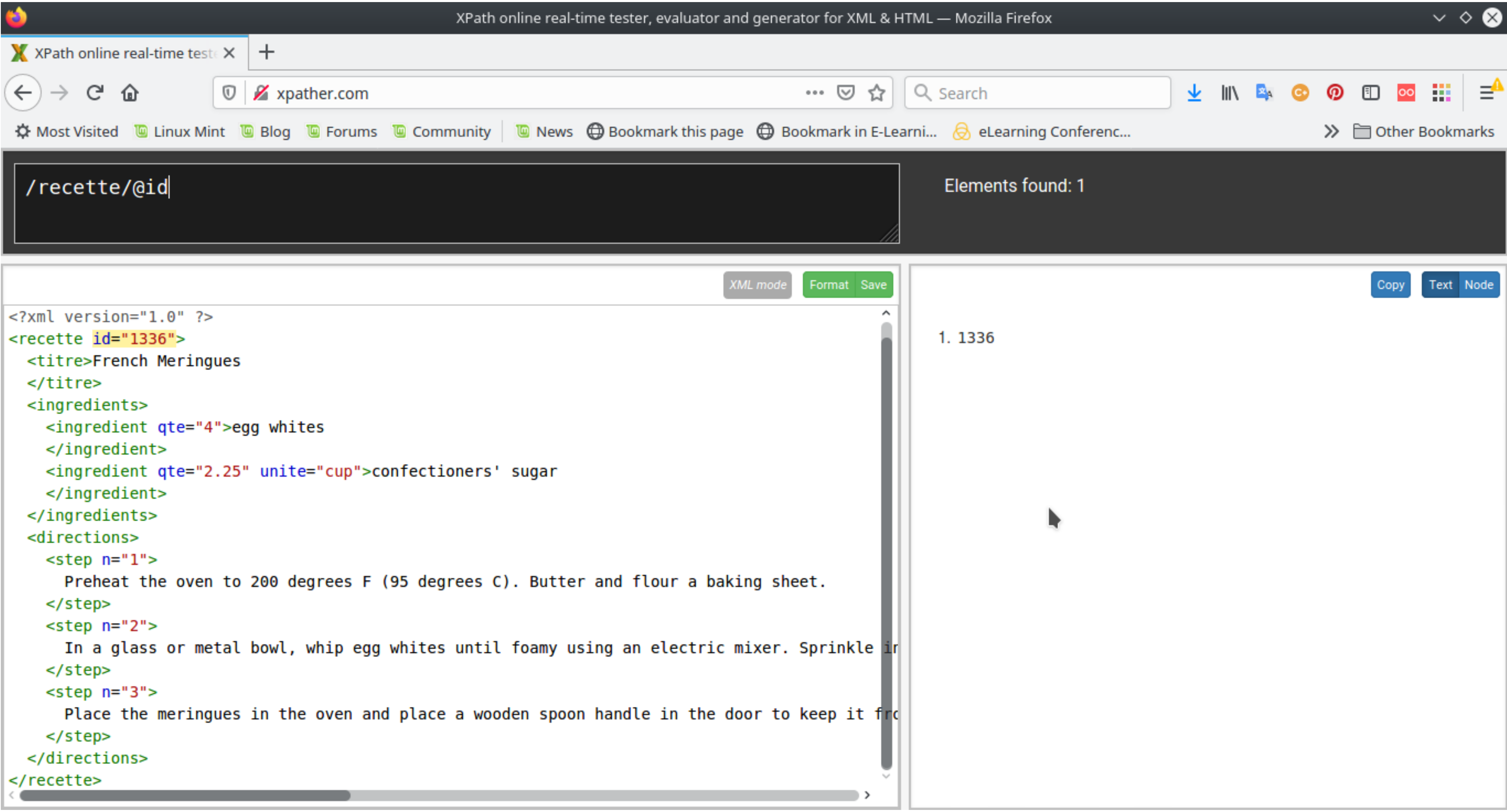
```
<?xml version="1.0" ?>
<recette id="1336">
  <titre>French Meringues
  </titre>
  <ingredients>
    <ingredient qte="4">egg whites
    </ingredient>
    <ingredient qte="2.25" unite="cup">confectioners' sugar
    </ingredient>
  </ingredients>
  <directions>
    <step n="1">
      Preheat the oven to 200 degrees F (95 degrees C). Butter and flour a baking sheet.
    </step>
    <step n="2">
      In a glass or metal bowl, whip egg whites until foamy using an electric mixer. Sprinkle ir
    </step>
    <step n="3">
      Place the meringues in the oven and place a wooden spoon handle in the door to keep it frc
    </step>
  </directions>
</recette>
```

```
1. egg whites
2. confectioners' sugar
```

XPath

- Example
  - Recipe.xml
    - Select the id of the recipe :
      - /recipe/@id

XPath



```
/recette/@id
```

Elements found: 1

```
<?xml version="1.0" ?>
<recette id="1336">
  <titre>French Meringues
  </titre>
  <ingredients>
    <ingredient qte="4">egg whites
    </ingredient>
    <ingredient qte="2.25" unite="cup">confectioners' sugar
    </ingredient>
  </ingredients>
  <directions>
    <step n="1">
      Preheat the oven to 200 degrees F (95 degrees C). Butter and flour a baking sheet.
    </step>
    <step n="2">
      In a glass or metal bowl, whip egg whites until foamy using an electric mixer. Sprinkle ir
    </step>
    <step n="3">
      Place the meringues in the oven and place a wooden spoon handle in the door to keep it frc
    </step>
  </directions>
</recette>
```

```
1. 1336
```

XPath

- Example
  - Predicates
    - To select nodes with specific values,
    - Always enclosed in square brackets [ ]

XPath

- Example
  - Predicates
    - position:
      - Select an element with the given position,
      - Elements are counted starting from 1,
      - Example: the second ingredient
        - //ingredient[2]

XPath

- Example
  - Predicates
    - last():
      - Select the last element
      - Example: select the last ingredient
        - //ingredient[last()]
    - position():
      - Returns the position of the element
      - Example: select the first two steps:
        - //step[position() < 3]

XPath

- Example
  - Predicates
    - @attribute:
      - Select the elements that have the attribute "attribute",
        - @attribute=value
    - Select the elements for which the attribute "attribute" has "value" as its value,
      - Example: select "step" number 2
        - //step[@n="2"]

XPath

- Example
  - Predicates
    - *: ALL
      - /*: all elements
      - /@*: all attributes

XPath

- Example
  - Select multiple paths:
    - |
    - Example: Select steps and ingredients:
      - //ingredient|//step

XPath

- Operators:
  - We have already used two operators:
    - =
    - <

XPath

- Operators :
  - Full list :

+	<
-	<=
*	>
div	>=
mod	and
=	or
!=	

Content

- DOM
- SAX
- XPATH
- **XSL**
- Exemple : SVG
- XLINK
- XPOINTER

XSL Transformation (XSLT)

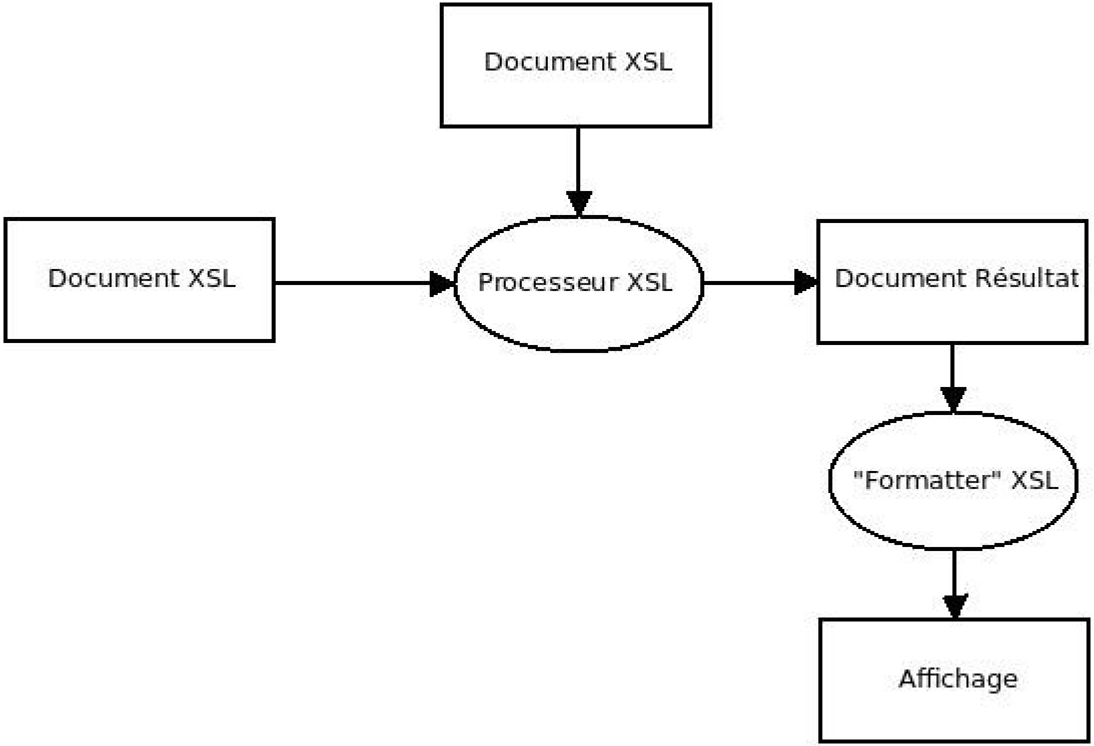
- Definition
  - Extensible Stylesheet Language
  - XSLT is a programming language (based on XML) that allows XML documents to be transformed into other formats.
  - It can also be used to transform one XML document into another (schema change).

XSL Transformation (XSLT)

- Definition
  - XSLT is a programming language (based on XML) that allows XML documents to be transformed into other formats.
  - It can also be used to transform one XML document into another (schema change).

XSL Transformation (XSLT)

- How they work:



XSL Transformation (XSLT)

- Link XML document to its XSL transformation sheet:
  - <?xml version = "1.0"?>
  - <?xml-stylesheet type = "text/xsl" href = "sheet.xsl"?>

XSL Transformation (XSLT)

- A basic XSLT file (with the nominal domain – namespace):
  - <?xml version = "1.0" ?>
  - <xsl:stylesheet
    - version = "1.0"
    - xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
  - >
  - </xsl:stylesheet>

XSL Transformation (XSLT)

- Basic elements:
  - "template"
    - XSLT allows you to define a template, that is, a model (or "mold") that will be populated from the XML file.
    - A template is applied to a pattern (XPath) specified in its definition.

XSL Transformation (XSLT)

- Basic elements:
  - “template”
    - <xsl:template
      - name = Qname
      - match = Pattern
      - priority = number
      - mode = QName
    - >
    - </xsl:template>

XSL Transformation (XSLT)

- Basic elements
  - "value-of"
    - Retrieve the value of an element from the XML file,
    - <xsl:value-of
      - select = Expression
    - >
    - </xsl:value-of>

XSL Transformation (XSLT)

- Basic elements:
  - "for-each"
    - Iterate (loop) to traverse a list of nodes,
    - <xsl:for-each
    - select = Expression >
    - </xsl:for-each>

XSL Transformation (XSLT)

- Basic elements:
  - "sort"
    - Sort the nodes (inside a loop, for example) according to a given criterion (value of an element, an attribute, etc.)
    - <xsl:sort
      - select = Expression
      - data-type = { "text" | "number" | QName }
      - order = { "ascending" | "descending" }
    - </xsl:sort>

XSL Transformation (XSLT)

- Basic elements:
  - "if"
    - The conditional structure: only nodes that meet a given condition will be processed.
    - <xsl:if
      - test="Boolean expression">
    - </xsl:if>
    - Example of a Boolean expression:
      - Value of an element or attribute compared using comparison operators.

XSL Transformation (XSLT)

- Basic elements:
  - "choose" with "when" and "otherwise"
    - Multiple conditional structures (equivalent to switch, case, and default in languages like C and Java).

XSL Transformation (XSLT)

- Basic elements:
  - "message"
    - Displays a message.
    - Can be used to express an error (execution stops).
    - <xsl:message
      - terminate = "yes" | "no" >
    - </xsl:message>
    - If "terminate" is set to "yes", execution stops and displays the message.

XSL Transformation (XSLT)

- Basic elements:
  - "apply-template"
    - It is possible to define several templates, each handling a specific pattern (path).
    - The main template will be responsible for calling the correct templates in the correct location according to the structure of the XML document.
    - The call is made through "apply-template".

XSL Transformation (XSLT)

- Basic elements:
  - "apply-template"
    - <xsl:apply-template
      - select = Expression
    - >
    - </xsl:apply-template>

XSL Transformation (XSLT)

- Example 01:
  - XML Data:
    - Products
      - Product
        - Label
        - Available Qty
        - Wholesale Price
        - Retail Price

XSL Transformation (XSLT)

- Example 01:
  - XML Data:
    - <products>
      - <product id="1">
        - <label>Blue Pen</label>
        - <available_quantity>1500</available_quantity>
        - <wholesale_price>7</wholesale_price>
        - <retail_price>15</retail_price>
      - </product>
      - …
    - </products>

XSL Transformation (XSLT)

- Example 01:
  - XML Transformation:
    - <xsl:stylesheet version="1.0"
    - xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      - <xsl:template match="**/**">
      - ...
      - </xsl:template>
    - </xsl:stylesheet>

XSL Transformation (XSLT)

- Example 01:
  - XML Transformation:
    - <html>
      - <body>
        - <h2>List of products</h2>
        - <table border="1">
          - <tr><th>Description</th> <th>Price</th></tr>
          - ...
        - </table>
      - </body>
    - </html>

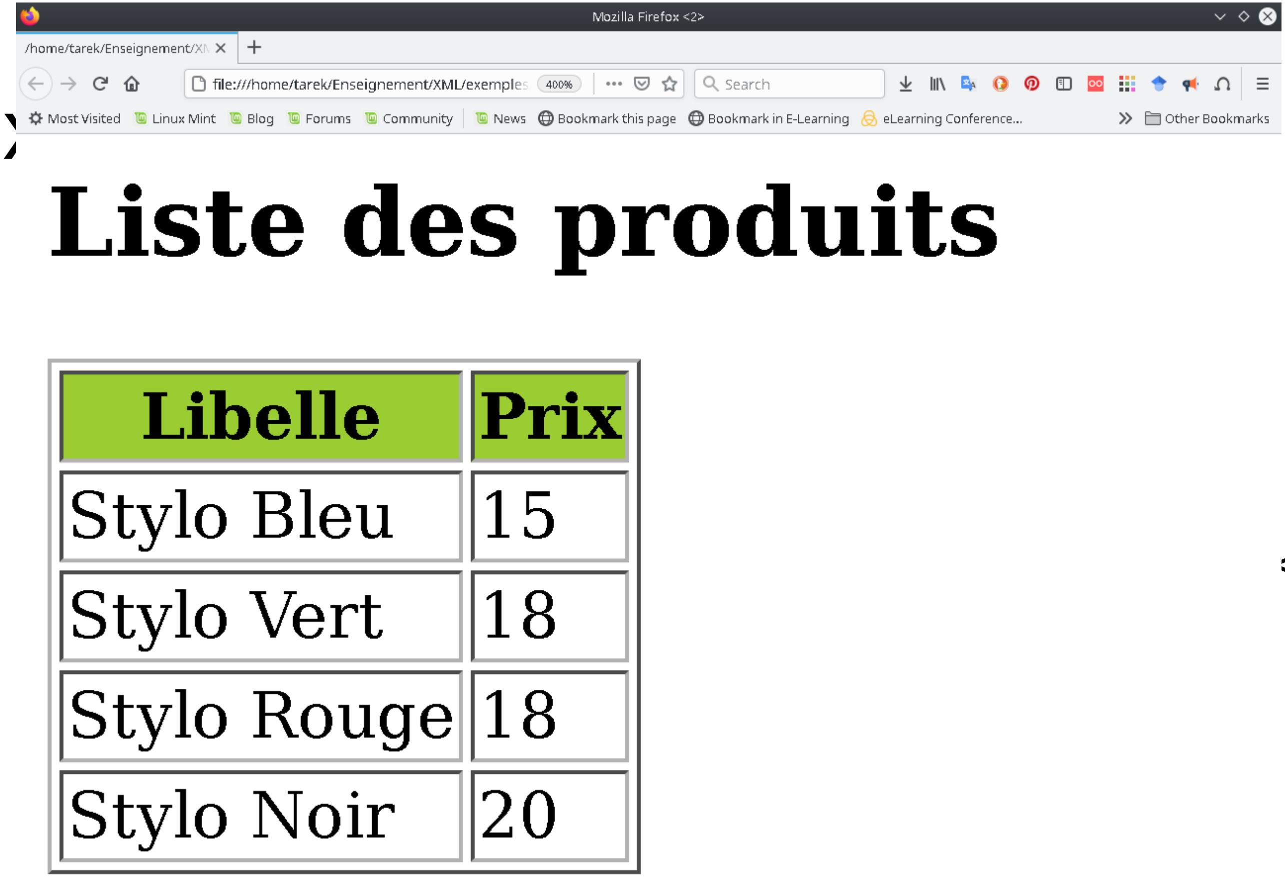
XSL Transformation (XSLT)

- Example 01:
  - XML Transformation:
    - <xsl:for-each select="**products/product**">
      - <tr>
        - <td><xsl:value-of select="**label**"/></td>
        - <td><xsl:value-of select="**retail_price**"/></td>
      - </tr>
    - </xsl:for-each>

XSL Transformation (XSLT)

- E

–



Liste des produits

Libelle	Prix
Stylo Bleu	15
Stylo Vert	18
Stylo Rouge	18
Stylo Noir	20

'>

</td>

XSL Transformation (XSLT)

- Example 02:
  - Temperatures:
    - Average for each month
    - All Wilayas
  - Only temperatures for Jijel are displayed.

XSL Transformation (XSLT)

- Example 02:
  - <temperatures>
    - <wilaya id="18">
      - <temperature month="January">14</temperature>
      - <temperature month="February">12</temperature>
      - <temperature month="March">19</temperature>
      - ...
    - </wilaya>
    - ...
  - </temperatures>

XSL Transformation (XSLT)

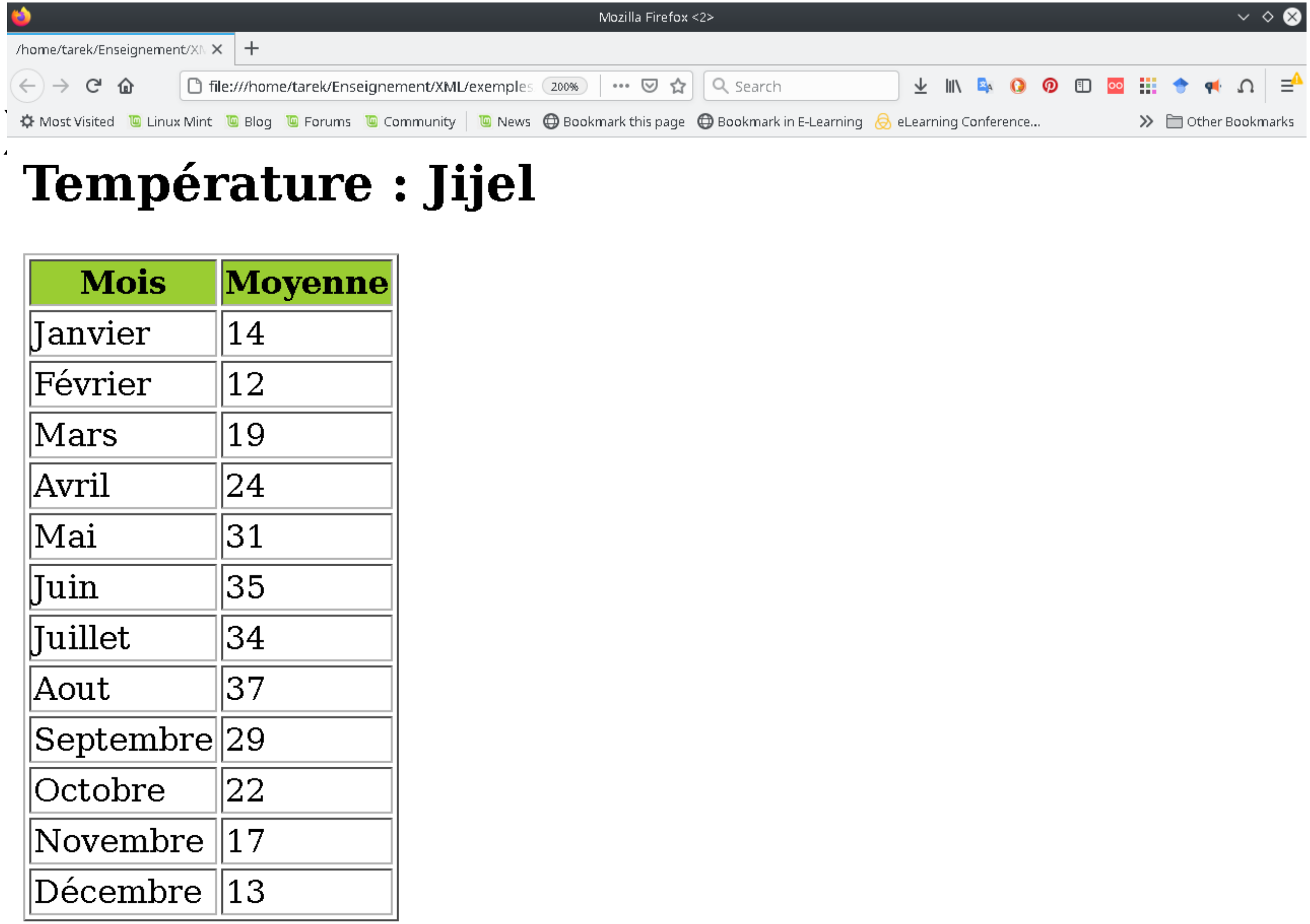
- Exemple 02 :
  - <xsl:template match="**/temperatures/wilaya[@id='18']**">
    - <html>
      - <body>
        - <table border="1">
          - <tr><th>Month</th><th>Average</th></tr>
          - ...
        - </table>
      - </body>
    - </html>
  - </xsl:template>

XSL Transformation (XSLT)

- Example 02:
  - <xsl:for-each select="**temperature**">
    - <tr>
      - <td><xsl:value-of select="**@month**"/></td>
      - <td><xsl:value-of select="**.**"/></td>
    - </tr>
  - </xsl:for-each>

XSL Transformation (XSLT)

- E
  - –
  - –



Température : Jijel

Mois	Moyenne
Janvier	14
Février	12
Mars	19
Avril	24
Mai	31
Juin	35
Juillet	34
Aout	37
Septembre	29
Octobre	22
Novembre	17
Décembre	13

XSL Transformation (XSLT)

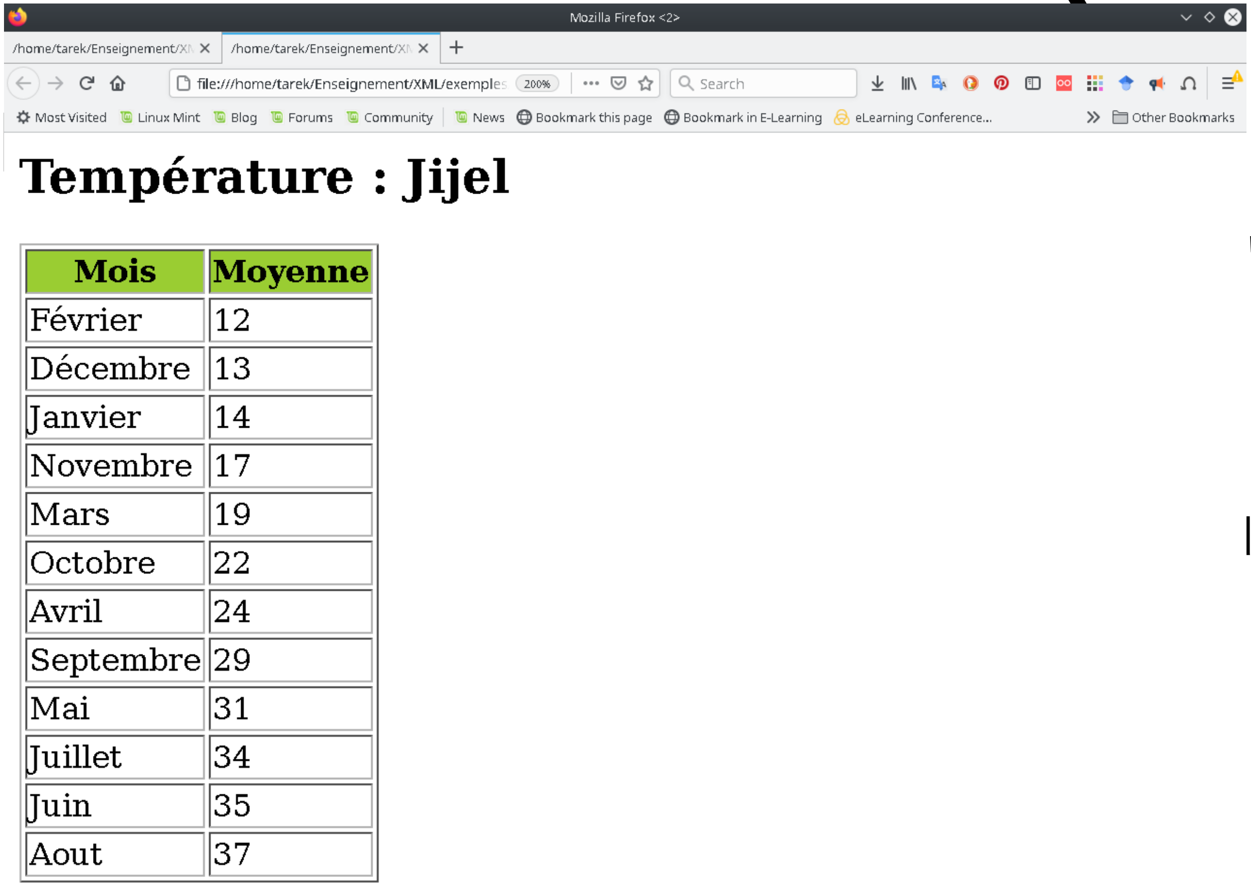
- Example 3:
  - Temperatures:
    - Average for each month
    - All Wilayas
  - Only temperatures for Jijel are displayed.
    - Months are sorted by temperature.

XSL Transformation (XSLT)

- Example 03:
  - <xsl:for-each select="**temperature**">
    - **<xsl:sort select="." />**
    - <tr>
      - <td><xsl:value-of select="**@month**"/></td>
      - <td><xsl:value-of select="**.**"/></td>
    - </tr>
  - </xsl:for-each>

XSL Transformation (XSLT)

-



Température : Jijel

Mois	Moyenne
Février	12
Décembre	13
Janvier	14
Novembre	17
Mars	19
Octobre	22
Avril	24
Septembre	29
Mai	31
Juillet	34
Juin	35
Aout	37

'>

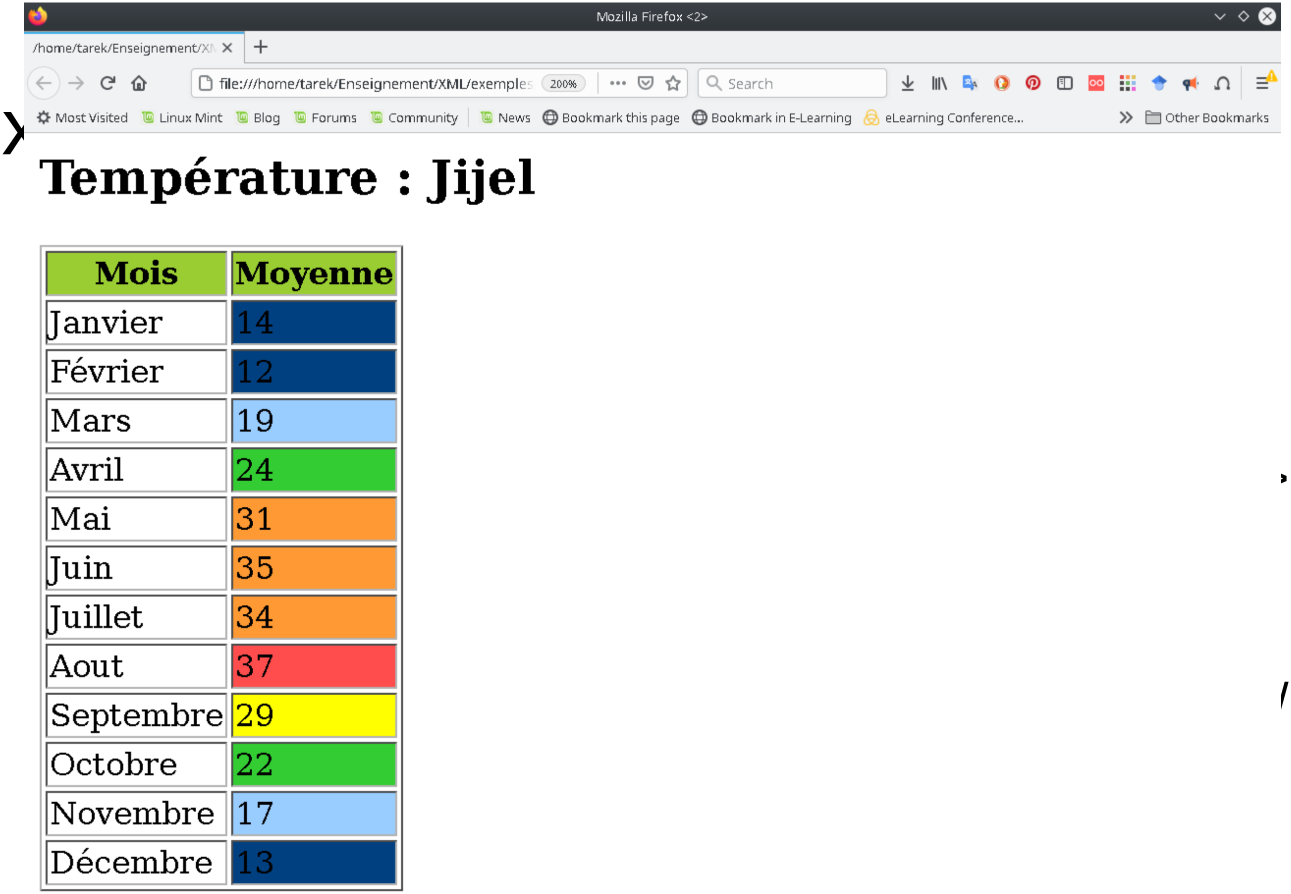
/>

XSL Transformation (XSLT)

- Example 04:
  - A little bit of decoration:
  - <xsl:choose>
    - <xsl:when test=". &gt; 35">
      - <td bgcolor="#ff4d4d"><xsl:value-of select="."/></td>
    - </xsl:when>
    - <xsl:otherwise>
      - <td bgcolor="#004080"><xsl:value-of select="."/></td>
    - </xsl:otherwise>
  - </xsl:choose>

XSL Transformation (XSLT)

- Ex
  -
  -



Température : Jijel

Mois	Moyenne
Janvier	14
Février	12
Mars	19
Avril	24
Mai	31
Juin	35
Juillet	34
Aout	37
Septembre	29
Octobre	22
Novembre	17
Décembre	13

</td>

/></td>

  -

Content

- DOM
- SAX
- XPATH
- XSL
- **Example : SVG**
- XLINK
- XPOINTER

SVG

- SVG
  - Scalable Vector Graphics
  - Allows you to define vector-based graphs using XML syntax.
  - Our goal:
    - Example in XML
    - We won't go into detail.

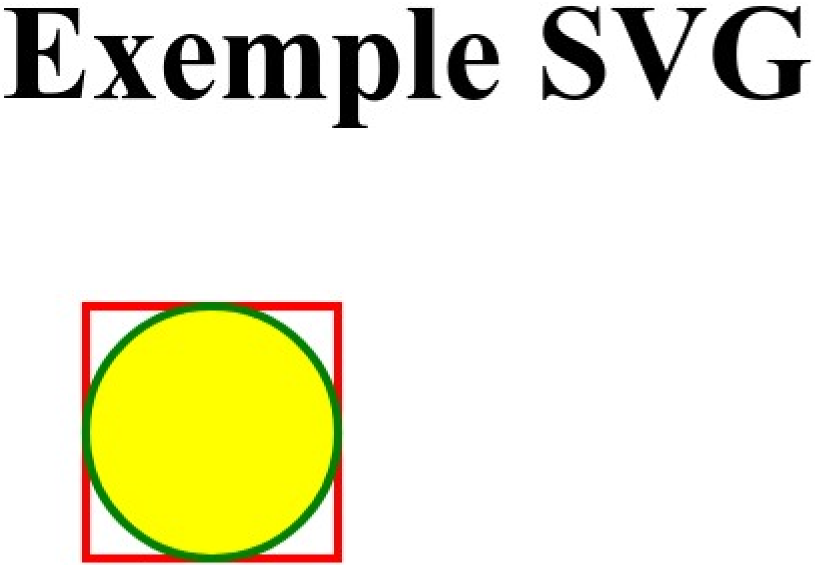
SVG

- Example:

```
 <html>
<body>

    <h1>SVG Example</h1>

    <svg width="100" height="100">
        <rect x="20" y="20"
             Height="60" width="60"
            stroke="red" stroke-width="2"
            fill="white" />
        <circle cx="50" cy="50"
             r="30"
             stroke="green" stroke-width="2"
             fill="yellow" />
    </svg>

</body>
</html>
```

SVG

- Example :



Exemple SVG

SVG

- SVG
  - List of forms (the simplest ones):
    - <rect>
    - <circle>
    - <ellipse>
    - <line>
    - <polyline>: a set of straight lines
    - <polygon>: closed polyline (the last point is linked to the first)

SVG

- SVG
  - <rect>
    - x, y: coordinates (top left point),
    - width, height: width and height,
    - rx, ry: rounding of corners.

SVG

- SVG
  - <circle>
    - cx, cy: coordinates (center),
    - r: radius

SVG

- SVG
  - <ellipse>
    - cx, cy: coordinates (center),
    - rx, ry: radii (horizontal, vertical).

SVG

- SVG
  - <line>
    - x1, y1: coordinates of the first point,
    - x2, y2: coordinates of the second point.

SVG

- SVG
  - <polyline>
    - points: list of points needed to draw the lines, set of x,y pairs:
      - points="100,100 150,25 150,75 200,0"

SVG

- SVG
  - <polygon>
    - points: list of points needed to draw the lines, set of x,y pairs:
      - points="100,100 150,25 150,75 200,0"

SVG

- SVG
  - Border and Fill:
    - Border:
      - stroke: border color, "none" to have no border,
      - stroke-width: border width,
      - stroke-dasharray: to make the border pointed:
        - stroke-dasharray = "4"
        - stroke-dasharray = "4 1"
    - stroke-linecap: define the shape of the end of the line
        - butt, round, square

SVG

- SVG
  - Border and Fill:
    - Fill:
      - fill: fill color, "none" to have no fill,
      - fill-opacity: fill transparency,
      - fill-rule: the algorithm defining the fill color
        - (nonzero, evenodd)

Content

- DOM
- SAX
- XPATH
- XSL
- Exemple : SVG
- **XLINK**
- XPOINTER

XLink

- Objective
  - To define a link (hyperlink),
  - The equivalent of the "a" tag in HTML,
  - Can be attached to any tag (no specific tag is required to define links)

XLink

- Usefulness
  - Defining Hypertext (and Hypermedia) structures by linking document elements to other resources on the Internet,
  - Annotating documents and resources,
  - Indexing documents and resources.

XLink

- Drawback
  - No precise tag,
  - Not supported by most browsers.

XLink

- Usage
  - Adding the namespace (nominal domain):
    - Prefix: xlink
    - URI: "http://www.w3.org/1999/xlink"
  - Adding attributes:
    - type (xlink:type)
      - To specify the link type.
    - Depending on this type, other attributes are added.

XLink

- Link Types
  - Simple
    - To specify that it is a simple link (similar to the "a" tag in HTML)
      - type = "simple"
    - Attributes:
      - href: link to the target,
      - role: role of the link,
      - title: title of the link,
      - ...

XLink

- Link Types
  - Extended
    - To point to multiple resources,
    - Type = "extended"
    - Attributes:
      - role: role of the link,
      - title: title of the link.

XLink

- Link Types
  - Locator
    - To point to an external resource,
    - Type = "locator"
    - Attributes:
      - href: the link to the resource,
      - role: the role of the link,
      - title: the title of the link.

XLink

- Link Types
  - Arc
    - To designate a passage (in the sense of crossing) from one resource to another,
    - Type = "arc"
    - Attributes:
      - title: link title,
      - from: starting resource,
      - to: destination resource.

XLink

- Link Types
  - Resource
    - To designate an internal resource,
    - Type = "resource"
    - Attributes:
      - title: link title,
      - role: of the link,
      - label: link label.

XLink

- Supported
  - XLink is not supported by most browsers.
  - Only the SVG extension includes simple links.
  - It is still possible to use them if a dedicated parser (processing program) is used.

Content

- DOM
- SAX
- XPATH
- XSL
- Exemple : SVG
- XLINK
- **XPOINTER**

XPointer

- Objective:
  - To point to a section of an XML document,
  - A more precise "link":
    - To a point, a node, or a region.
- Principle:
  - Combination of:
    - XLink: to point to the document,
    - XPath: to point to the section.

XPointer

- Syntax:
  - A link (xlink:href),
  - A clarification (Xpointer expression) after the symbol: #
  - Example:
    - xlink:href="http://univ-jijel.dz/doc.xml#xpointer(id('cours_01')"

XPointer

- XPointer Expressions
  - Access by ID
    - Function: id()
    - id("value"), or more simply:
    - Value
      - http://univ-jijel.dz/doc.xml#cours_01

XPointer

- XPointer Expressions
  - Child Sequence
    - Using the position of the nodes.
    - Function: element()
    - Example: the second child, of the third child, of the element with the id cours_01:
      - element(cours_01/3/2)

XPointer

- XPointer Expressions
  - Other functions:
    - self()
    - origin()
    - here()

XPointer

- XPointer Expressions
  - Block-based expression construction:
    - child Locates: child nodes of the selected node,
    - descendant: node inside the selected node,
    - descendant-or-self: like "descendant" but includes the selected node,
    - parent: the parent node (direct higher level) of the selected node,
    - ancestor: all nodes above the selected node,
    - ancestor-or-self: like "ancestor" but includes the selected node.

XPointer

- XPointer Expressions
  - Block-based expression construction:
    - preceding-sibling: previous "sibling",
    - following-sibling: next "sibling",
    - preceding: previous nodes,
    - following: following nodes,
    - attribute: access to the attributes of the selected node.

University of Jijel
Faculty of Exact Sciences and Computer Science
Department of Computer Science
L3 – Computer Systems

Semi-Structured Data
Chapter 4
XML Galaxy

Tarek Boutefara
t_boutefara@univ-jijel.dz

2025/2026