Réécriture pour la programmation et la preuve

Horatiu CIRSTEA, Serguei LENGLET, Pierre-Etienne MOREAU

XML - Extensible Markup Language

- a Meta-Markup Language
- designed to describe data - a structural and semantic language, not a formatting language
- tags are not predefined - you must define your own tags
- a W3C Recommendation

Used for

- Exchange Data: data can be exchanged between incompatible systems.
- Share Data: data is stored in plain text format
- Store Data
- Create new Languages

Syntax

```
<master>
  <student>
    <name>Asterix</name>
    <rank over="120">14</rank>
    <note><!-- Over 20 -->
    12
    </note>
  </student>

  <student gender="male">
    <name>Obelix</name>
    <rank over="120">12</rank>
    <note><!-- Over 20 -->
    15
    </note>
  </student>
</master>
```

Syntax

- All XML elements must have a closing tag

```
<name>Asterix</name>
```

- XML tags are case sensitive

- All XML elements must be properly nested

```
<student gender="male">
  <name>Asterix
  </student>
</name>
```

- All XML documents must have a root element

```
<master>
...
</master>
```

Syntax

- All other elements must be within this root element.

- Attribute values must always be quoted

```
<rank over="120">14</rank>
<name nickn='the "best"'>Asterix</name>
<name nickn='the &quot;best&quot;'>Asterix</name>
```

- Spaces are preserved

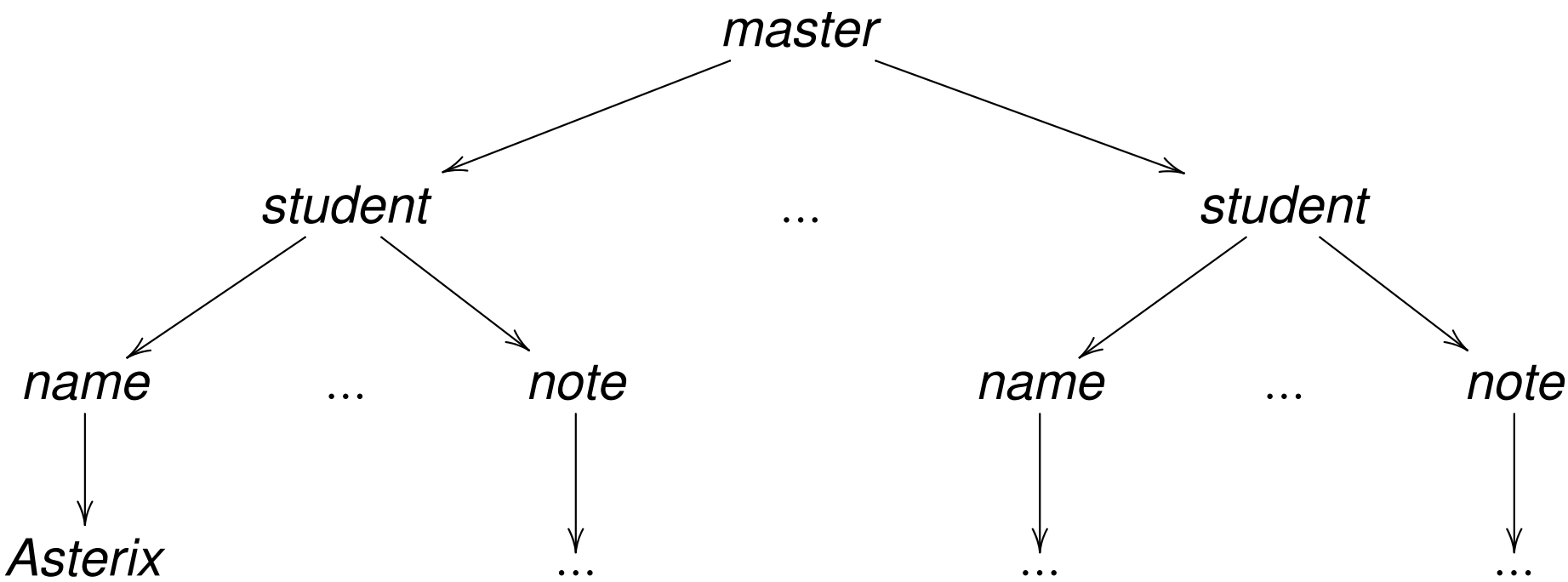
- $<$ and $>$ are used only for tags
  (& is used as escape character in entities:
  &*amp*, &*lt*, &*gt*, &*quot*, &*apos*)

```
<master>
Math &amp; Info
...
</master>
```

Documents as trees

Any XML document can be seen as a tree

- a root element
- children (of different types)



Complete XML documents

- the prolog
- the root element (and nested elements)
- processing instructions
- comments

The prolog

- XML declaration

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

- document type declaration

```
<!DOCTYPE master [...]>
<!DOCTYPE master SYSTEM "lmd.dtd">
<!DOCTYPE university PUBLIC "-//Webuniv//DTD University 1.0//EN"
"http://www.webuniv.com/university/DTD/lmd.dtd">
```

- processing instructions

```
<?xml-stylesheet type="text/css" href="master.css" ?>
<?xml-stylesheet type="text/xml" href="master.xsl" ?>
```

- comments

```
<!- A comment ->
```

View XML documents

- XML editors: Eclipse, Oxygen, ...
- Browsers - generally display only text elements
- Get a better display
  - use style sheets
  - use XSLT

View XML documents

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<master>
  <student gender="male">
    <name>Asterix</name>
    <rank over="120">14</rank>
    <note><!-- Over 20 -->
    12
    </note>
  </student>

  <student gender="male">
    <name>Obelix</name>
    <rank over="120">12</rank>
    <note><!-- Over 20 -->
    15
    </note>
  </student>
</master>
```

Browser (Firefox) display

Asterix 14 12 Obelix 12 15

Using Style Sheets

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/css" href="master.css" ?>

<master>
  <student>
    <name>Asterix</name>
    <rank over="120">14</rank>
    <note><!-- Over 20 -->
    12
    </note>
  </student>

  <student gender="male">
    <name>Obelix</name>
    <rank over="120">12</rank>
    <note><!-- Over 20 -->
    15
    </note>
  </student>
</master>
```

Style Sheet Example

```
name   {display: block
          font-family: Helvetica, sans-serif;
          font-size: 20pt; font-weight: bold;}

rank   {display: block; }

note   {display: block;
          font-family: Times, "Times New Roman", serif;
          font-style: italic;}
```

Browser (Firefox) improved display

Asterix
14
12
Obelix
12
15

XSL Transformations

Extensible Stylesheet Language

Three parts

- a formatting language (XSL-FO)

Extensible Stylesheet Language

Three parts

- a formatting language (XSL-FO)
- a transformation language (XSLT)

Extensible Stylesheet Language

Three parts

- a formatting language (XSL-FO)
- a transformation language (XSLT)
- a data model and query language (XPath)

The transformation process

XSLT transforms one XML tree into another XML tree:

- The XML document transformed into a tree
- The XSLT processor matches the nodes in the tree against the rules in the style sheet
- When a match is found, the corresponding tree is output

Nodes of an XML tree

- The root
- Elements
- Attributes
- Text
- Processing instructions
- Comments
- Namespaces

Simplest XSLT stylesheet

```
<?xml version="1.0"?>
<xsl:stylesheet  version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
</xsl:stylesheet>
```

Apply on XML file

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE master SYSTEM "master.dtd">
<master>
  <student>
    <name>Asterix</name>
    <rank over="120"> 14 </rank>
    <note> 12 </note>
  </student>

  <student gender="male">
    <name>Obelix</name>
    <rank over="120"> 12 </rank>
    <note> 15 </note>
  </student>
</master>
```

Result (`saxon -v -o simple.html master.xml simple.xsl`):

```
<?xml version="1.0" encoding="UTF-8"?>Asterix 14  12 Obelix 12  15
```

How are the XML documents transformed w.r.t. a stylesheet?

- The client (Web browser) transforms the document as specified by the stylesheet and presents it to the user.
- The server applies the stylesheet to the XML document to transform it to some other format (generally HTML) and sends the transformed document to the client (Web browser).
- A program (e.g. `saxon`) transforms the original XML document into some other format (often HTML) and the result is placed on the server.

XML file handled by the Web browser

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE master SYSTEM "lmd.dtd">

<?xml-stylesheet type="text/xml" href="master2.xsl" ?>

<master>
  <student gender="male">
    <name>Asterix</name>
    <rank over="120">14</rank>
    <note><!-- Over 20 -->    12    </note>
  </student>

  <student gender="male">
    <name>Obelix</name>
    <rank over="120">12</rank>
    <note><!-- Over 20 -->    15    </note>
  </student>
</master>
```

A simple template - xsl:template

```
<?xml version="1.0"?>
<xsl:stylesheet  version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
    <html><title>Master students</title>
       <body>
       </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```

Result (`saxon -v -o master1.html master.xml master1.xsl`):

```
<html>
   <title>Master students</title>
   <body></body>
</html>
```

A (template) rule

- `xsl:template`: element that represents a rule
- The `match` pattern is an `XPath` expression tried against the elements of the XML tree starting with the root node of the document. For example,
  - match the root: `match="/"`
  - match an element: `match="name"`
  - match any element: `match="*"`
- If the pattern is matched, the template (i.e. the content of the `xsl:template` element) is instantiated (and added to the complete result tree).
- Some default rules apply when no explicit rule matches.

Select the value of a node - xsl:value-of

```
<?xml version="1.0"?>
<xsl:stylesheet  version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="ISO-8859-1"  />

<xsl:template match="name">
    <h3><xsl:value-of select="."/></h3>
</xsl:template>

</xsl:stylesheet>
```

Select the value of a node - `xsl:value-of`

```
<?xml version="1.0"?>
<xsl:stylesheet  version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="ISO-8859-1"  />

<xsl:template match="name">
    <h3><xsl:value-of select="."/></h3>
</xsl:template>

</xsl:stylesheet>
```

Result (`saxon -v -o master2.html master.xml master2.xsl`):

```
<h3>Asterix</h3> 14  12
<h3>Obelix</h3> 12  15
```

Output the value of a node

- `xsl:value-of`: output the value of a node
- The `select` pattern is an `XPath` expression used to select the node(s) to be output. For example,
  - select the current node: `select="."`
  - select an element: `select="name"`
  - select an attribute: `select="@over"`
- The value of a node depends on its type

Output a node value - `xsl:value-of`

```
<xsl:stylesheet  version="2.0"   xmlns:xsl="http://www.w3.c

<xsl:template match="student">
    <h3><xsl:value-of select="name"/></h3>
</xsl:template>

</xsl:stylesheet>
```

Computing the node value - xsl:value-of

```
<xsl:stylesheet  version="2.0"   xmlns:xsl="http://www.w3.c

<xsl:template match="student">
    <h3><xsl:value-of select="name"/></h3>
</xsl:template>

</xsl:stylesheet>
```

```
<h3>Asterix</h3>
<h3>Obelix</h3>
```

Going down the XML tree - xsl:apply-templates

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org

<xsl:template match="/">
    <xsl:apply-templates />
</xsl:template>
<xsl:template match="master">
    <xsl:apply-templates />
</xsl:template>

<xsl:template match="student">
    <h3>Student</h3>
</xsl:template>

</xsl:stylesheet>
```

Going down the XML tree - xsl:apply-templates

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org

<xsl:template match="/">
    <xsl:apply-templates />
</xsl:template>
<xsl:template match="master">
    <xsl:apply-templates />
</xsl:template>

<xsl:template match="student">
    <h3>Student</h3>
</xsl:template>

</xsl:stylesheet>
```

```
<h3>Student</h3>
<h3>Student</h3>
```

Same behaviour with implicit rules

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org

<xsl:template match="*|/">
    <xsl:apply-templates/>
</xsl:template>

<xsl:template match="student">
    <h3>Student</h3>
</xsl:template>

</xsl:stylesheet>
```

```
<h3>Student</h3>
<h3>Student</h3>
```

Apply templates recursively

- `xsl:apply-templates`: apply all the templates recursively on all child (element or text) nodes or
- on nodes obtained with a `select` pattern which is an `XPath` expression. For example,
  - select a text element: `select="text()"`
  - select any attribute: `select="@*"`

Recall: output a node value with `xsl:value-of`

```
<xsl:stylesheet  version="2.0"   xmlns:xsl="http://www.w3.c

<xsl:template match="student">
    <h3><xsl:value-of select="name"/></h3>
</xsl:template>

</xsl:stylesheet>
```

```
<h3>Asterix</h3>
<h3>Obelix</h3>
```

Alternative - `select` the names

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org

<xsl:template match="student">
  <h3>
    <xsl:apply-templates select="name"/>
  </h3>
</xsl:template>

</xsl:stylesheet>
```

```
<h3>Asterix</h3>
<h3>Obelix</h3>
```

Why does it work? Another implicit rule

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org

<xsl:template match="text()|@*">
    <xsl:value-of select="."/>
</xsl:template>

<xsl:template match="student">
  <h3>
    <xsl:apply-templates select="name"/>
  </h3>
</xsl:template>

</xsl:stylesheet>
```

```
<h3>Asterix</h3>
<h3>Obelix</h3>
```

Output the (text) information in all elements

```
<?xml version="1.0"?>
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org

<xsl:template match="/">
    <html><title>Master students</title>
       <body>
         <xsl:apply-templates/>
       </body>
    </html>
</xsl:template>

</xsl:stylesheet>
```

```
<html>
   <title>Master students</title>
   <body>Asterix 14  12 Obelix 12  15 </body>
</html>
```

Exercise: output names with rank and note

```
<h3>Asterix</h3>
<p>Rank:  14 </p>
<p>Note:  12 </p>
<h3>Obelix</h3>
<p>Rank:  12 </p>
<p>Note:  15 </p>
```

Exercise: output names with with rank and note

```
<xsl:stylesheet  version="2.0"   xmlns:xsl="http://www.w3.org/1999/XSL/Tr

<xsl:template match="name">
    <h3><xsl:value-of select="."/></h3>
</xsl:template>
<xsl:template match="rank">
    <p>Rank: <xsl:value-of select="."/></p>
</xsl:template>
<xsl:template match="note">
    <p>Note: <xsl:value-of select="."/></p>
</xsl:template>

</xsl:stylesheet>
```

```
<h3>Asterix</h3>
<p>Rank:  14 </p>
<p>Note:  12 </p>
<h3>Obelix</h3>
<p>Rank:  12 </p>
<p>Note:  15 </p>
```

Exercise: output the relative rank for each name

```
<h3>Asterix</h3> 14  / 120
<h3>Obelix</h3> 12  / 120
```

Exercise: output the relative rank for each name

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:template match="student">
  <h3>
    <xsl:apply-templates select="name"/>
  </h3>
  <xsl:apply-templates select="rank"/>
</xsl:template>

<xsl:template match="rank">
    <xsl:value-of select="."/> / <xsl:value-of select="@over"/>
</xsl:template>

</xsl:stylesheet>
```

```
<h3>Asterix</h3> 14  / 120
<h3>Obelix</h3> 12  / 120
```

More complex output: generate attributes in output

```
<h3>Asterix</h3> 14  /<a href="http://www.ranks.com/120.html">120</a>

<h3>Obelix</h3> 12  /<a href="http://www.ranks.com/120.html">120</a>
```

More complex output: generate attributes in output

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:template match="student">
<xsl:text>
</xsl:text>
  <h3>
    <xsl:apply-templates select="name"/>
  </h3>
  <xsl:apply-templates select="rank"/>
</xsl:template>

<xsl:template match="rank">
 <xsl:value-of select="."/> /<a href="http://www.ranks.com/{@over}.html">
 <xsl:value-of select="@over"/>  </a>
</xsl:template>

</xsl:stylesheet>
```

```
<h3>Asterix</h3> 14  /<a href="http://www.ranks.com/120.html">120</a>

<h3>Obelix</h3> 12  /<a href="http://www.ranks.com/120.html">120</a>
```

More patterns

- **root node**

```
<xsl:template match="/">...</xsl:template>
```

- **element node**

```
<xsl:template match="student">...</xsl:template>
```

- **wildcard**

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/

<xsl:template match="master">
    <xsl:apply-templates />
</xsl:template>

<xsl:template match="*">
    <p><xsl:value-of select="."/></p>
</xsl:template>

</xsl:stylesheet>
```

```
<p>Asterix 14  12 </p>
<p>Obelix 12  15 </p>
```

licence-master file

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE university SYSTEM "lmd.dtd">

<university>


<master>
  <manager>
    <name>Abraracourcix</name>
  </manager>

  <student>
    <name>Asterix</name>
    <rank over="120">14</rank>
    <note>12</note>
  </student>

  <student gender="male">
    <name>Obelix</name>
    <rank over="120">12</rank>
    <note>15</note>
  </student>
</master>

<licence>
  <student>
    <name>Abraracourcix</name>
    <rank over="140">27</rank>
    <note>11</note>
  </student>

  <student gender="male">
    <name>Bonemine</name>
    <rank over="140">3</rank>
    <note>16</note>
  </student>
</licence>

</university>
```

Matching children - all the students

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org

<xsl:template match="student/name">
    <p><xsl:value-of select="."/></p>
</xsl:template>

</xsl:stylesheet>
```

```
Abraracourcix<p>Obelix</p> 12  15
<p>Asterix</p> 14  12
<p>Bonemine</p> 109  11
<p>Agécanonix</p> 11  7
```

Matching children - master and licence students

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:template match="master/manager"></xsl:template>

<xsl:template match="master/student">
   <font color="blue"><p><xsl:value-of select="name"/></p></font>
</xsl:template>

<xsl:template match="licence/student">
   <font color="red"><p><xsl:value-of select="name"/></p></font>
</xsl:template>

</xsl:stylesheet>
```

```
<font color="blue">
   <p>Obelix</p></font><font color="blue">
   <p>Asterix</p></font><font color="red">
   <p>Bonemine</p></font><font color="red">
   <p>Agécanonix</p></font>
```

Matching children with wildcards - don't print managers

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:template match="/*/*/manager"></xsl:template>

<xsl:template match="master/student">
   <p><xsl:value-of select="name"/></p>
</xsl:template>

<xsl:template match="licence/student">
   <p><xsl:value-of select="name"/></p>
</xsl:template>

</xsl:stylesheet>
```

```
<p>Obelix</p>
<p>Asterix</p>
<p>Bonemine</p>
<p>Agécanonix</p>
```

Matching descendants - all the master persons

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:template match="licence"></xsl:template>

<xsl:template match="master//name">
   <p><xsl:value-of select="."/></p>
</xsl:template>

</xsl:stylesheet>
```

```
<p>Abraracourcix</p>
<p>Obelix</p> 12  15
<p>Asterix</p> 14  12
```

Alternatives - get rid of the notes and ranks

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:template match="licence"></xsl:template>

<xsl:template match="note | rank"></xsl:template>

<xsl:template match="master//name">
   <p><xsl:value-of select="."/></p>
</xsl:template>

</xsl:stylesheet>
```

```
<p>Abraracourcix</p>
<p>Obelix</p>
<p>Asterix</p>
```

Matching atributes - relative notes and ranks

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran
<xsl:template match="licence"></xsl:template>
<xsl:template match="manager/name"></xsl:template>
<xsl:template match="master/student">
   <p><xsl:value-of select="name"/></p>
   <p><xsl:apply-templates select="rank"/></p>
</xsl:template>
<xsl:template match="rank">
 <font color="red"><xsl:value-of select="."/></font>/
 <xsl:apply-templates select="@over"/>
</xsl:template>
<xsl:template match="@over">
   <font color="blue"><xsl:value-of select="."/></font>
</xsl:template>
</xsl:stylesheet>
```

```
<p>Obelix</p>
<p><font color="red"> 12 </font>/
   <font color="blue">120</font></p>
<p>Asterix</p>
<p><font color="red"> 14 </font>/
   <font color="blue">120</font></p>
```

Matching (differently) atributes in descendants

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran
<xsl:template match="name"></xsl:template>
<xsl:template match="rank"></xsl:template>

<xsl:template match="note">
 <p><font color="red"><xsl:value-of select="."/></font>/
 <xsl:apply-templates select="@over"/></p>
</xsl:template>

<xsl:template match="licence//*/@over">
   <font color="green"><xsl:value-of select="."/></font>
</xsl:template>
<xsl:template match="master//*/@over">
   <font color="blue"><xsl:value-of select="."/></font>
</xsl:template>
</xsl:stylesheet>

<p><font color="red"> 15 </font>/
   <font color="blue">20</font></p>
<p><font color="red"> 12 </font>/
   <font color="blue">20</font></p>
<p><font color="red"> 11 </font>/
   <font color="green">20</font></p>
<p><font color="red"> 7 </font>/
   <font color="green">20</font></p>
```

master_with_comment file

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE master SYSTEM "lmd.dtd">

<?xml-stylesheet type="text/xml" href="master2.xsl" ?>

<master>
  <student gender="male">
    <name>Asterix</name>
    <rank over="120">14</rank>
    <note><!-- Over 20 -->    12    </note>
  </student>

  <student gender="male">
    <name>Obelix</name>
    <rank over="120">12</rank>
    <note><!-- Over 20 -->    15    </note>
  </student>
</master>
```

Matching comments

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran
<xsl:template match="name"></xsl:template>
<xsl:template match="rank"></xsl:template>

<xsl:template match="note">
 <p><font color="red"><xsl:value-of select="."/></font>
 <xsl:apply-templates select="comment()"/></p>
</xsl:template>

<xsl:template match="note/comment()">
<i><xsl:value-of select="."/></i>
</xsl:template>

</xsl:stylesheet>
```

```
<p><font color="red">    12    </font><i> Over 20 </i></p>
<p><font color="red">    15    </font><i> Over 20 </i></p>
```

Matching explicitly `text()` nodes

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org

<xsl:template match="licence"></xsl:template>

<xsl:template match="note | rank"></xsl:template>

<xsl:template match="text()">
   <p><b><xsl:value-of select="."/></b></p>
</xsl:template>

</xsl:stylesheet>
```

```
<p><b>Abraracourcix</b></p>
<p><b>Obelix</b></p>
<p><b>Asterix</b></p>
```

XPATH

XSLT

- language for addressing parts of an XML document.
- not an XML language.
- an expression language that allows the processing of values conforming to a (tree) data model.
- expressions evaluated in a (dynamic) context (`http://www.w3.org/TR/2007/REC-xpath20-20070123`):
  - the context item
  - the context position
  - the context size
  - variable values
  - function implementations
  - ...

XSLT expressions

- Primary Expressions
- Path Expressions
- Sequence Expressions
- Arithmetic Expressions
- Comparison Expressions
- Logical Expressions
- Conditional Expressions
- ...

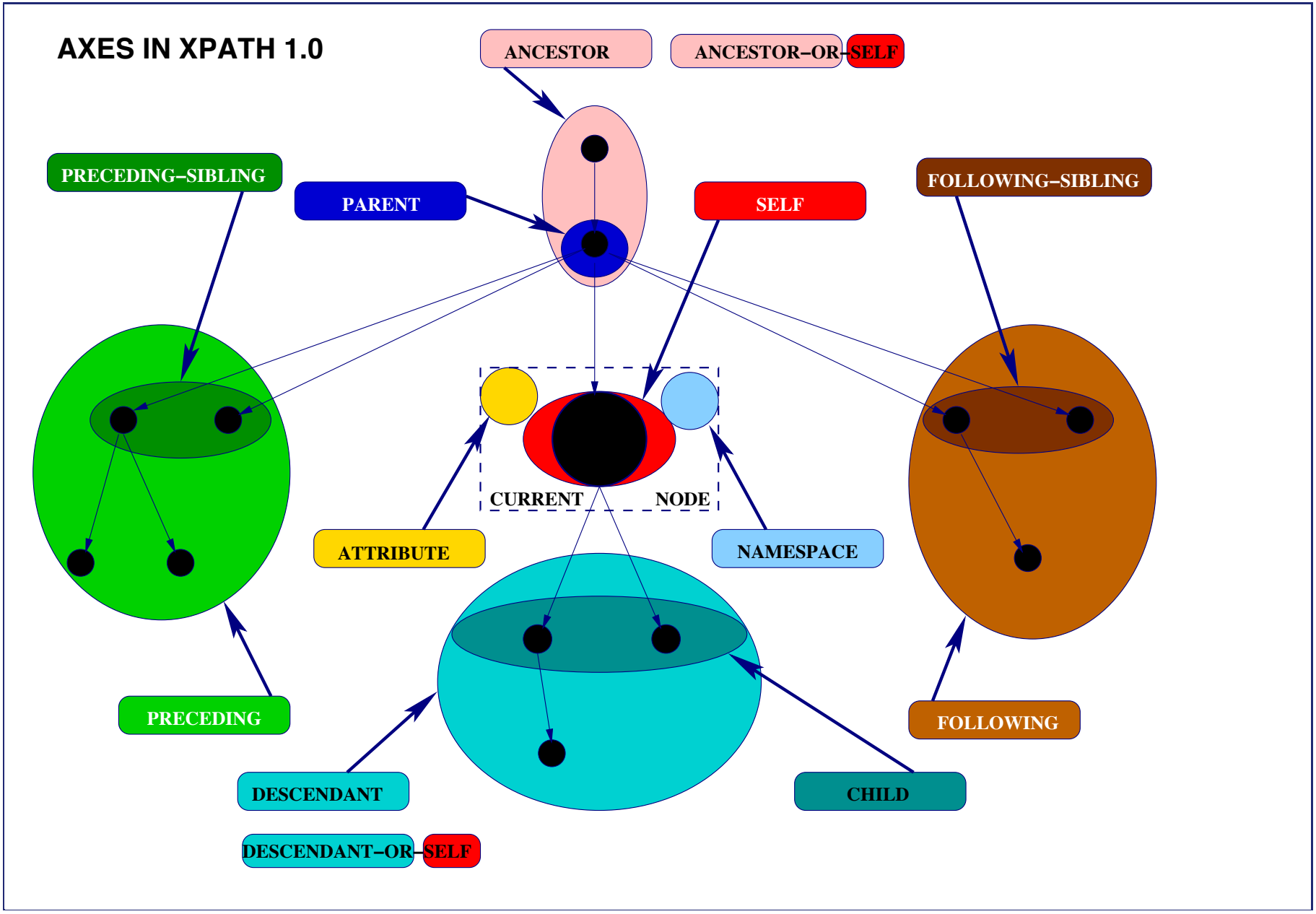
Path Expressions

- Constructed from steps (`axis::node-test[predicate]`):
  - Axes
  - Node Tests
  - Predicates
- Can use
  - unabbreviated syntax
  - abbreviated syntax
- Can be
  - absolute: `/master/student`
  - relative: `student/name`

XSLT axes

- self - the Context Node
- child - children of the CN
- parent - parent of the CN
- ancestor(-or-self) - ancestors of the CN (and the CN)
- descendant(-or-self) - descendants of the CN (and the CN)
- following(-sibling) - (sibling) nodes that follow the CN
- preceding(-sibling) - (sibling) nodes that precede the CN
- attribute - attributes of the CN

XSLT axes (from Dan Olteanu)



Students with TOEFL - licence-master-toefl.xml

```
<master>
  <manager gender="male">
    <name>Abraracourcix</name>
  </manager>

  <student>
    <name>Asterix</name>
    <rank over="120"> 14 </rank>
    <note> 12 </note>


    <toefl> 15 </toefl>


  </student>
  <student gender="male">
    <name>Obelix</name>
    <rank over="120"> 12 </rank>
    <note over="20"> 15 </note>
  </student>
</master>
```

```
<licence>
  <student>
    <name>Agecanonix</name>
    <rank over="140"> 27 </rank>
    <note> 11 </note>
  </student>
  <student gender="female">
    <name>Bonemine</name>
    <rank over="140"> 3 </rank>
    <note over="20"> 16 </note>


    <toefl> 13 </toefl>


  </student>
</licence>
```

Students with TOEFL notes - select descendants and parent

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:template match="master | licence">
    <xsl:apply-templates
select="self::node()/descendant-or-self::node()/child::toefl"/>
</xsl:template>

<xsl:template match="toefl">
   <p><b><xsl:value-of select="parent::student"/></b></p>
</xsl:template>

</xsl:stylesheet>
```

```
<p><b>Asterix 14  12  15 </b></p>
<p><b>Bonemine 3  16  13 </b></p>
```

Abbreviations

`_`	`child::`
`.`	`self::node()`
`..`	`parent::node()`
`@_`	`attribute::`
`.//`	`./descendant-or-self::node()/`
`//`	`descendant-or-self::node()/`
`*`	all children
`@*`	all attributes
`[n]`	`[position() = n]`

Students with TOEFL notes - use abbreviations

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org

<xsl:template match="master | licence">
    <xsl:apply-templates  select=".//toefl"/>
</xsl:template>

<xsl:template match="toefl">
   <p><b><xsl:value-of select="parent::student"/></b></p>
</xsl:template>

</xsl:stylesheet>
```

```
<p><b>Asterix 14  12  15 </b></p>
<p><b>Bonemine 3  16  13 </b></p>
```

Select current node and attributes - print the name of students with TOEFL notes

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:template match="master | licence">
    <xsl:apply-templates  select=".//toefl"/>
</xsl:template>

<xsl:template match="toefl">
<p><xsl:value-of select="parent::*/child::name"/>:
<xsl:value-of select="self::*"/> / <xsl:value-of select="attribute::over"
</p>
</xsl:template>

</xsl:stylesheet>
```

```
<p>Asterix:
    15  / 20
</p>
<p>Bonemine:
    13  / 20
</p>
```

Alternative - use the `siblings`

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:template match="master | licence">
    <xsl:apply-templates  select=".//toefl"/>
</xsl:template>

<xsl:template match="toefl">
   <p><xsl:value-of select="preceding-sibling::name |
                            following-sibling::name"/>:
   <xsl:value-of select="self::*"/> /
     <xsl:value-of select="attribute::over"/>
   </p>
</xsl:template>
</xsl:stylesheet>


<p>Asterix:
    15  /
   20
</p>
<p>Bonemine:
    13  /
   20
</p>
```

XPath built-in functions (http://www.w3.org/TR/xquery-operators/)

- Functions and Operators on Numerics
  - Operators: `op:numeric-add`, `op:numeric-subtract`,...
  - Functions: `abs`, `ceiling`,`round`,...
- Functions on Strings
  - Equality and Comparison of Strings: `compare`, `codepoint-equal`,...
  - Functions on String Values `concat`, `substring`, `upper-case`,...
  - Functions Based on Substring Matching `contains`, `starts-with`,...
  - ...

XPath built-in functions (http://www.w3.org/TR/xquery-operators/)

- Functions and Operators on Boolean Values
  - Constructors: `true`, `false`
  - Operators on Boolean Values: `equal`,...
  - Functions on Boolean Values: `not`
- Context Functions: `position`, `last`, ...
- Functions and Operators on Durations, Dates and Times
- Functions and Operators on Nodes
- ...

Example - functions on context lists

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:template match="master | licence">
   <xsl:apply-templates select="student"/>
</xsl:template>

<xsl:template match="student[position() &lt; 2]">
 <p><b><xsl:value-of select="position()"/>/<xsl:value-of select="last()"/
       <xsl:value-of select="name"/></p>
</xsl:template>

</xsl:stylesheet>
```

```
<p><b>1/2. </b>Obelix
</p>Asterix 14  12
<p><b>1/2. </b>Bonemine
</p>Agécanonix 11  7
```

Example - precise position

```
<xsl:template match="master | licence">
   <xsl:apply-templates select="student[1]"/>
</xsl:template>

<xsl:template match="student">
 <p><b><xsl:value-of select="position()"/>/<xsl:value-of select="last()"/
       <xsl:value-of select="name"/></p>
</xsl:template>

</xsl:stylesheet>
```

Example - check attributes

```
<xsl:template match="master | licence">
   <xsl:apply-templates select="student"/>
</xsl:template>

<xsl:template match="student[@gender]">
 <p><xsl:value-of select="name"/>(<xsl:value-of select="@gender"/>)</p>
</xsl:template>

<xsl:template match="student[not(@gender)]">
 <p><xsl:value-of select="name"/>(???)</p>
</xsl:template>

</xsl:stylesheet>
```

Example - compute average notes

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:template match="master">
   Average Master= <xsl:value-of select="sum(descendant::note)
                                  div count(descendant::note)"/>
</xsl:template>

<xsl:template match="licence">
   Average Licence = <xsl:value-of select="sum(descendant::note)
                                  div count(descendant::note)"/>
</xsl:template>


</xsl:stylesheet>
```

```
Average Master= 13.5
Average Licence = 9
```

Example - functions on strings

```
<xsl:template match="master | licence">
   <xsl:apply-templates select="student[contains(name,'x')]"/>
</xsl:template>
```

More examples

- nodes without attributes:
  `//*[not(@*)]`
- note of the second master student:
  `master//student[2]/note`
- student with good marks:
  `//student[note > 13]`
- student with average marks:
  `//student[note > 9 and note < 14]`
- level with 3 students:
  `//(licence | master)[count(.//student) = 2]`
- first student from each level:
  `//(licence | master)//student[1]`
- first student in university:
  `(//(licence | master)//student)[1]`

More examples

- students whose name contains “ix”:
  `//student[contains(name,'ix')]`
- level with all student names containing “ix”:
  `//*[count(.//student[contains(name,'ix')]) = count(.//student) and .//student]`
- student whose note is unique:
  `//student[not(note = preceding::student/note) and not(note = following::student/note)]`
- unique students:
  `//student[not(. = preceding::student) and not(. = following::student)]`

Programming

Flow control

- looping
- conditionals
- sorting

Looping

- Use `<xsl:for-each select=”...”>...<xsl:for-each>`
- The `select` pattern is an `XPath` expression used to select the set of nodes.
- The contents of the `xsl:for-each` is output one time for each node in the set.

Iterate over elements with `xsl:for-each`

```
<xsl:stylesheet  version="2.0"   xmlns:xsl="http://www.w3.org/1999/XSL/Tr

<xsl:template match="master">
   <xsl:for-each select="student">
       <h3><xsl:value-of select="name"/></h3>
       <p>Rank: <xsl:value-of select="rank"/></p>
       <p>Note: <xsl:value-of select="note"/></p>
   </xsl:for-each>
</xsl:template>

</xsl:stylesheet>
```

```
<h3>Asterix</h3>
<p>Rank:  14 </p>
<p>Note:  12 </p>
<h3>Obelix</h3>
<p>Rank:  12 </p>
<p>Note:  15 </p>
```

Performing tests

- Use `<xsl:if test="expression">...</xsl:if>`
- The `expression` is an `XPath` expression and may contain
  - String expressions: `concat(e1,e2), substring(e1),...`
  - Boolean expressions: `false(), true(), not(be), e1 = e2, e1 != e2, ...`
  - Numeric expressions: `number(e) ne1 op ne2, count(ne), sum(ne), round(ne), position(), last(),...`
  - NodeSet expressions

Perform tests with `xsl:if`

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran
<xsl:output method="html" encoding="ISO-8859-1" doctype-public="-//W3C//D

<xsl:template match="student">
   <xsl:if test="number(note)>13">
    <h3>
      <xsl:apply-templates select="name"/>
    </h3>
  </xsl:if>
</xsl:template>


</xsl:stylesheet>
```

```
<h3>Obelix</h3>
```

Choose with `xsl:choose`

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:template match="student">
   <xsl:apply-templates select="name"/>
   <xsl:choose>
        <xsl:when test="number(note)  &lt; 10">
          <font color="red"> <xsl:value-of select="note"/></font>
        </xsl:when>
        <xsl:when test="number(note)  &gt; 9 and number(note)  &lt; 15">
          <font color="black"> <xsl:value-of select="note"/></font>
        </xsl:when>
        <xsl:otherwise>
          <font color="blue"> <xsl:value-of select="note"/></font>
        </xsl:otherwise>
   </xsl:choose>
 <xsl:text>
 </xsl:text>
</xsl:template>


</xsl:stylesheet>
```

Choose with `xsl:choose`

```
Asterix<font color="black"> 12 </font>
 Obelix<font color="blue"> 15 </font>
```

Matching tests

- Value of element
- Matching value of element against string
- Existence of element
- Position of element

Male managers and female students

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:template match="student"></xsl:template>
<xsl:template match="manager"></xsl:template>

<xsl:template match="student[@gender!='male']">
   <p><b><xsl:value-of select="name"/></b></p>
</xsl:template>

<xsl:template match="manager[@gender='male']">
   <p><b><xsl:value-of select="name"/></b></p>
</xsl:template>

</xsl:stylesheet>
```

```
<p><b>Abraracourcix</b></p>
<p><b>Bonemine</b></p>
```

Print gender if available

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:template match="student">
   <p><b><xsl:value-of select="name"/></b></p>
</xsl:template>
<xsl:template match="*[@gender]">
   <p><b><xsl:value-of select="name"/></b>
 (<xsl:value-of select="@gender"/>)</p>
</xsl:template>

</xsl:stylesheet>
```

```
<p><b>Abraracourcix</b>
     (male)
</p>
<p><b>Obelix</b>
     (male)
</p>
<p><b>Asterix</b></p>
<p><b>Bonemine</b>
     (female)
</p>
<p><b>Agécanonix</b></p>
```

Sorting

- Use `<xsl:sort order="..." data-type="..." select="..." />`
- The `expression` is an `XPath` expression used to identify the node to sort by.
- The `data-type` specifies the type of the node: text, number, ...
- The `order` specifies: ascending, descending.

Sort the students by name and rank

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:template match="master">
    <xsl:apply-templates select="student">
       <xsl:sort select="name" />
    </xsl:apply-templates>
</xsl:template>
<xsl:template match="licence">
    <xsl:apply-templates select="student">
       <xsl:sort order="descending" data-type="number" select="rank" />
    </xsl:apply-templates>
</xsl:template>

<xsl:template match="student">
   <p><b><xsl:value-of select="name"/></b></p>
   <xsl:value-of select="rank"/>
</xsl:template>
</xsl:stylesheet>
```

```
<p><b>Asterix</b></p> 14
<p><b>Obelix</b></p> 12
<p><b>Bonemine</b></p> 109
<p><b>Agécanonix</b></p> 11
```

Create and copy nodes

- elements
- attributes
- comments

Create comments

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran
<xsl:template match="name"></xsl:template>
<xsl:template match="rank"></xsl:template>

<xsl:template match="note">
 <p><font color="red"><xsl:value-of select="."/></font>
 <xsl:apply-templates select="comment()"/></p>
</xsl:template>

<xsl:template match="note/comment()">
   <xsl:comment><xsl:value-of select="."/></xsl:comment>
</xsl:template>

</xsl:stylesheet>
```

```
<p><font color="red">    12    </font>
   <!-- Over 20 --></p>
<p><font color="red">    15    </font>
   <!-- Over 20 --></p>
```

More general - create elements

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:template match="/">
     <xsl:for-each select="//student">
          <xsl:element name="{@gender}">
               <xsl:value-of select="."/>
          </xsl:element>
     </xsl:for-each>
</xsl:template>

</xsl:stylesheet>
```

```
<male>Asterix14    12    </male>
<male>Obelix12    15    </male>
```

More general - create elements and attributes

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:template match="/">
     <xsl:for-each select="//student">
          <xsl:element name="{@gender}">
               <xsl:attribute name="rank">
                         <xsl:value-of select="rank"/>
                         <xsl:text>-over-20</xsl:text>
               </xsl:attribute>
               <xsl:value-of select="name"/>
          </xsl:element>
     </xsl:for-each>
</xsl:template>

</xsl:stylesheet>
```

```
<male rank="14-over-20">Asterix</male>
<male rank="12-over-20">Obelix</male>
```

Copy elements - get rid of the names

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:template match="/">
     <xsl:for-each select="//student">
          <xsl:copy>
              <xsl:copy-of select="name"/>
          </xsl:copy>
        </xsl:for-each>
</xsl:template>

</xsl:stylesheet>
```

```
<student>
   <name>Asterix</name>
</student>
<student>
   <name>Obelix</name>
</student>
```

Copy nodes - all the students in one university

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:template match="university">
    <xsl:element name="{./master/manager/name}">
       <xsl:apply-templates />
    </xsl:element>
</xsl:template>

<xsl:template match="master | licence">
       <xsl:apply-templates  select="student"/>
</xsl:template>

<xsl:template match="student">
   <xsl:copy-of  select="." />
</xsl:template>

</xsl:stylesheet>
```

Copy nodes - all the students in one university

```
<Abraracourcix>
   <student>
      <name>Asterix</name>
      <rank over="120"> 14 </rank>
      <note over="20"> 12 </note>
      <toefl over="20"> 15 </toefl>
   </student>
   <student gender="male">
      <name>Obelix</name>
      <rank over="120"> 12 </rank>
      <note over="20"> 15 </note>
   </student>
   <student>
      <name>Agécanonix</name>
      <rank over="140"> 27 </rank>
      <note over="20"> 11 </note>
   </student>
   <student gender="female">
      <name>Bonemine</name>
      <rank over="140"> 3 </rank>
      <note over="20"> 16 </note>
      <toefl over="20"> 13 </toefl>
   </student>
</Abraracourcix>
```

Copy nodes - all the TOEFL students in one university

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:template match="university">
    <xsl:element name="{@name}">
       <xsl:apply-templates />
    </xsl:element>
</xsl:template>

<xsl:template match="master | licence">
       <xsl:apply-templates  select=".//toefl"/>
</xsl:template>

<xsl:template match="toefl">
   <xsl:copy-of  select=".." />
</xsl:template>

</xsl:stylesheet>
```

Copy nodes

```
<?xml version="1.0" encoding="UTF-8"?>

<Nancy>

  <student>
    <name>Asterix</name>
    <rank over="120"> 14 </rank>
    <note over="20"> 12 </note>
    <toefl over="20"> 15 </toefl>
  </student>

  <student gender="female">
    <name>Bonemine</name>
    <rank over="140"> 3 </rank>
    <note over="20"> 16 </note>
    <toefl over="20"> 13 </toefl>
  </student>

</Nancy>
```

Advanced templates

- different modes
- named templates
- parametrized templates
- variables

Different rules for the same element

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:template match="university">
    <xsl:apply-templates select="*/student" mode="list" >
       <xsl:sort select="name" />
    </xsl:apply-templates>
    <xsl:apply-templates select="master/student" mode="completeMaster" />
    <xsl:apply-templates select="licence/student" mode="completeLicence"
</xsl:template>

<xsl:template match="student" mode="list">
   <xsl:value-of select="name"/>/
</xsl:template>

<xsl:template match="student" mode="completeMaster">
   <p><font color="red"><xsl:value-of select="name"/></font>
   <xsl:value-of select="rank"/></p>
</xsl:template>
<xsl:template match="student" mode="completeLicence">
   <p><font color="blue"><xsl:value-of select="name"/></font>
   <xsl:value-of select="rank"/></p>
</xsl:template>

</xsl:stylesheet>
```

Different rules for the same element

```
Agécanonix/
Asterix/
Bonemine/
Obelix/
<p><font color="red">Obelix</font> 12
</p>
<p><font color="red">Asterix</font> 14
</p>
<p><font color="blue">Bonemine</font> 109
</p>
<p><font color="blue">Agécanonix</font> 11
</p>
```

xsl:variables

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:variable name="header">
Students of the  <xsl:value-of select="/university/@name"/>
</xsl:variable>

<xsl:template match="master">
    <xsl:value-of select="$header"/> master
    <hr />
    <xsl:apply-templates select="student">
       <xsl:sort select="name" />
    </xsl:apply-templates>
</xsl:template>

<xsl:template match="licence">
    <xsl:value-of select="$header"/> licence
    <hr />
    <xsl:apply-templates select="student">
       <xsl:sort select="name" />
    </xsl:apply-templates>
</xsl:template>

<xsl:template match="student">
   <p><b><xsl:value-of select="name"/></b>
   <xsl:value-of select="rank"/></p>
</xsl:template>
```

xsl:variables

```
Students of the  Nancy Universities master
<hr>
<p><b>Asterix</b> 14
</p>
<p><b>Obelix</b> 12
</p>
Students of the  Nancy Universities licence

<hr>
<p><b>Agécanonix</b> 11
</p>
<p><b>Bonemine</b> 109
</p>
```

xsl:variables and named xsl:templates

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:variable name="header">
Students of the  <xsl:value-of select="/university/@name"/>
</xsl:variable>

<xsl:template name="student">
    <xsl:apply-templates select="student">
       <xsl:sort select="name" />
    </xsl:apply-templates>
</xsl:template>

<xsl:template match="master">
    <xsl:value-of select="$header"/>: master
    <hr />
    <xsl:call-template name="student"/>
</xsl:template>

<xsl:template match="licence">
    <xsl:value-of select="$header"/>: licence
    <hr />
    <xsl:call-template name="student"/>
</xsl:template>

<xsl:template match="student">
   <p><b><xsl:value-of select="name"/></b>
```

Named templates with parameters

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:template name="student">
  <xsl:param name="level"></xsl:param>
  <xsl:value-of select="$level"/> students:
  <hr />
    <xsl:apply-templates select="student">
       <xsl:sort select="name" />
    </xsl:apply-templates>
</xsl:template>

<xsl:template match="master">
    <xsl:call-template name="student">
       <xsl:with-param name="level">Master</xsl:with-param>
    </xsl:call-template>
</xsl:template>

<xsl:template match="licence">
    <xsl:call-template name="student">
       <xsl:with-param name="level">Licence</xsl:with-param>
    </xsl:call-template>
</xsl:template>

<xsl:template match="student">
   <p><b><xsl:value-of select="name"/></b>
   <xsl:value-of select="rank"/></p>
```

Named templates with parameters

```
Master students:
<hr>
<p><b>Asterix</b> 14
</p>
<p><b>Obelix</b> 12
</p>Licence students:

<hr>
<p><b>Agécanonix</b> 11
</p>
<p><b>Bonemine</b> 109
</p>
```

Named templates with parameters

```
<xsl:stylesheet  version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Tran

<xsl:template name="student">
  <xsl:param name="level"></xsl:param>
     <xsl:choose>
          <xsl:when test="$level='Master'">
               Master<HR/>
               <xsl:apply-templates select="//master/student"/>
          </xsl:when>

          <xsl:when test="$level='Licence'">
               Licence<HR/>
               <xsl:apply-templates select="//licence/student"/>
          </xsl:when>

          <xsl:otherwise>
               <HR/>
               <xsl:apply-templates select="//student"/>
          </xsl:otherwise>
     </xsl:choose>
</xsl:template>

<xsl:template match="/">
    <xsl:call-template name="student">
       <xsl:with-param name="level">Master</xsl:with-param>
    </xsl:call-template>
```

Named templates with parameters

```
<xsl:template match="/">
    <xsl:call-template name="student">
       <xsl:with-param name="level">Master</xsl:with-param>
    </xsl:call-template>
    <xsl:call-template name="student">
       <xsl:with-param name="level">Licence</xsl:with-param>
    </xsl:call-template>
    <xsl:call-template name="student">
       <xsl:with-param name="level">All</xsl:with-param>
    </xsl:call-template>
</xsl:template>

<xsl:template match="student">
   <p><b><xsl:value-of select="name"/></b>
   <xsl:value-of select="rank"/></p>
</xsl:template>
</xsl:stylesheet>
```