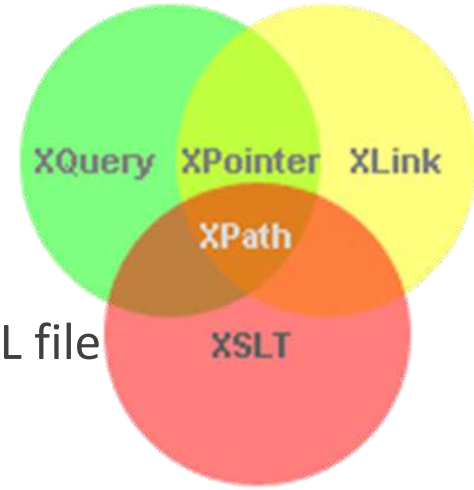
XSL

DR WIOLETA SZWOCH

DEPARTMENT OF INTELLIGENT INTERACTIVE SYSTEMS

XPath (*XML Path Language*)



standard for identifying parts of an XML document

An unambiguous description of the element's address in the XML file

Used in other standards, does not exist alone

Path expressions are used to navigate in XML documents

- syntax similar to Unix file system paths
- the ability to extract the necessary nodes

XPath contains a library of standard functions

standard per identificare parti di un documento XML
Una descrizione univoca dell'indirizzo dell'elemento nel file XML
Utilizzato in altri standard, non esiste da solo

Le espressioni di percorso vengono utilizzate per navigare nei documenti XML:
- sintassi simile ai percorsi del file system Unix
- la capacità di estrarre i nodi necessari
XPath contiene una libreria di funzioni standard

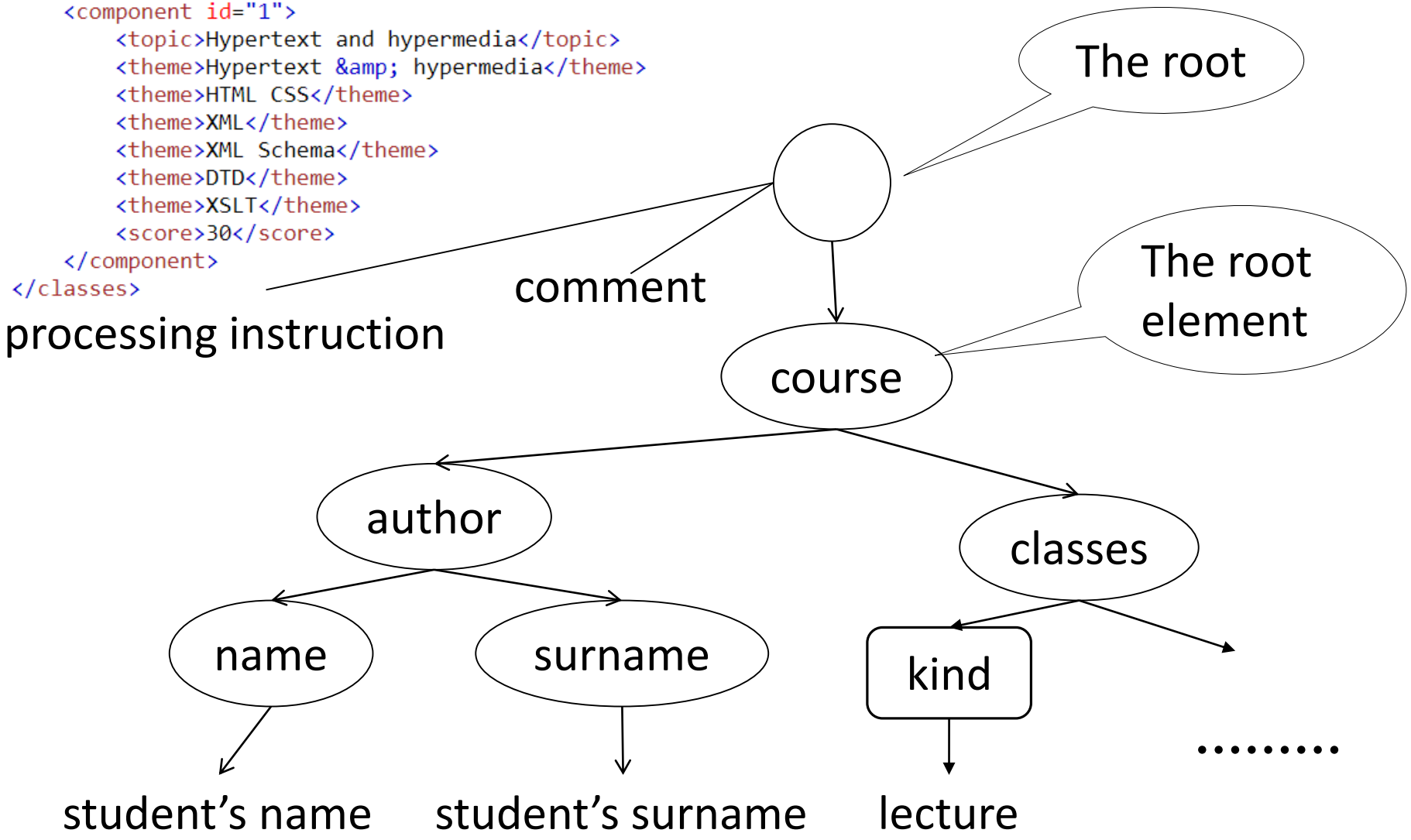
Nodes

XML document

- Tree structure with nodes
- Each node represents part of XML document
  - Seven types
    - Root
    - Element
    - Attribute
    - Text
    - Comment
    - Processing instruction
    - Namespace

Data model for XPath

```
<course>
    <author>
        <name>student's name</name>
        <surname>student's surname</surname>
    </author>
    <classes kind="lecture" >
        <component id="1">
            <topic>Hypertext and hypermedia</topic>
            <theme>Hypertext &amp;amp; hypermedia</theme>
            <theme>HTML CSS</theme>
            <theme>XML</theme>
            <theme>XML Schema</theme>
            <theme>DTD</theme>
            <theme>XSLT</theme>
            <score>30</score>
        </component>
    </classes>
```



```xml
<course>
    <author>
        <name>student's name</name>
        <surname>student's surname</surname>
    </author>
    <classes kind="lecture" >
        <component id="1">
            <topic>Hypertext and hypermedia</topic>
            <theme>Hypertext &amp;amp; hypermedia</theme>
            <theme>HTML CSS</theme>
            <theme>XML</theme>
            <theme>XML Schema</theme>
            <theme>DTD</theme>
            <theme>XSLT</theme>
            <score>30</score>
        </component>
    </classes>
```

XPath is a syntax for "addressing" into a document.
They are "path expressions".
XPath expressions have a directory-path-like syntax.

relative path
step/step/...
absolute path
/step/step/...

A step consists of:
- an axis
- a node-test
- zero or more predicates

Step – full syntax:
axis::**node-test** [**predicate1**] [**predicate2**] ...

- **axis** – direction in document tree
- **node-test** – selecting nodes by kind, name, or type
- **predicates** – (0 or more) additional logical conditions for filtering

```xml
<course>
    <author>
        <name>student's name</name>
        <surname>student's surname</surname>
    </author>
    <classes kind="lecture" >
        <component id="1">
            <topic>Hypertext and hypermedia</topic>
            <theme>Hypertext &amp;amp; hypermedia</theme>
            <theme>HTML CSS</theme>
            <theme>XML</theme>
            <theme>XML Schema</theme>
            <theme>DTD</theme>
            <theme>XSLT</theme>
            <score>30</score>
        </component>
    </classes>
```

XPath is a syntax for "addressing" into a document.
They are "path expressions".
XPath expressions have a directory-path-like syntax.

"/" represents the Document info item (root)

* matches any element — author/*

@ means attribute — classes/@kind

```
<course>
    <author>
        <name>student's name</name>
        <surname>student's surname</surname>
    </author>
    <classes kind="lecture" >
        <component id="1">
            <topic>Hypertext and hypermedia</topic>
            <theme>Hypertext &amp;amp; hypermedia</theme>
            <theme>HTML CSS</theme>
            <theme>XML</theme>
            <theme>XML Schema</theme>
            <theme>DTD</theme>
            <theme>XSLT</theme>
            <score>30</score>
        </component>
    </classes>
```

classes//theme //theme

The result of evaluating an XPath expression is a Node Set

"//" matches elements that aren't direct children

XPath

XPath treat XML as a tree of elements

Root of tree – document node main element (aka document element) is not the root

„Leaf” may be:

tag, attribute, processing instruction, comment, text, namespace

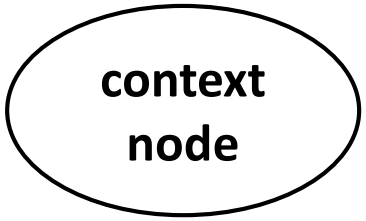
”leaves” are related by ”branches” - axes

Nodes

Relations between nodes

child, parent, descendant , ancestor, sibling

```xml
<course>
    <author>
        <name>student's name</name>
        <surname>student's surname</surname>
    </author>
    <classes kind="lecture" >
        <component id="1">
            <topic>Hypertext and hypermedia</topic>
            <theme>Hypertext &amp;amp; hypermedia</theme>
            <theme>HTML CSS</theme>
            <theme>XML</theme>
            <theme>XML Schema</theme>
            <theme>DTD</theme>
            <theme>XSLT</theme>
            <score>30</score>
        </component>
    </classes>
```



You can address the context node as '**.**'

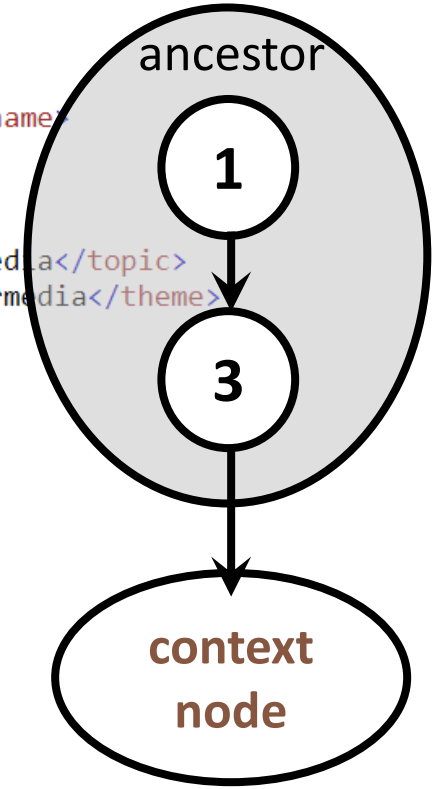
./@*

The context node is implicit.

range/component ≡ ./range/component

The context node does not have to be an element.

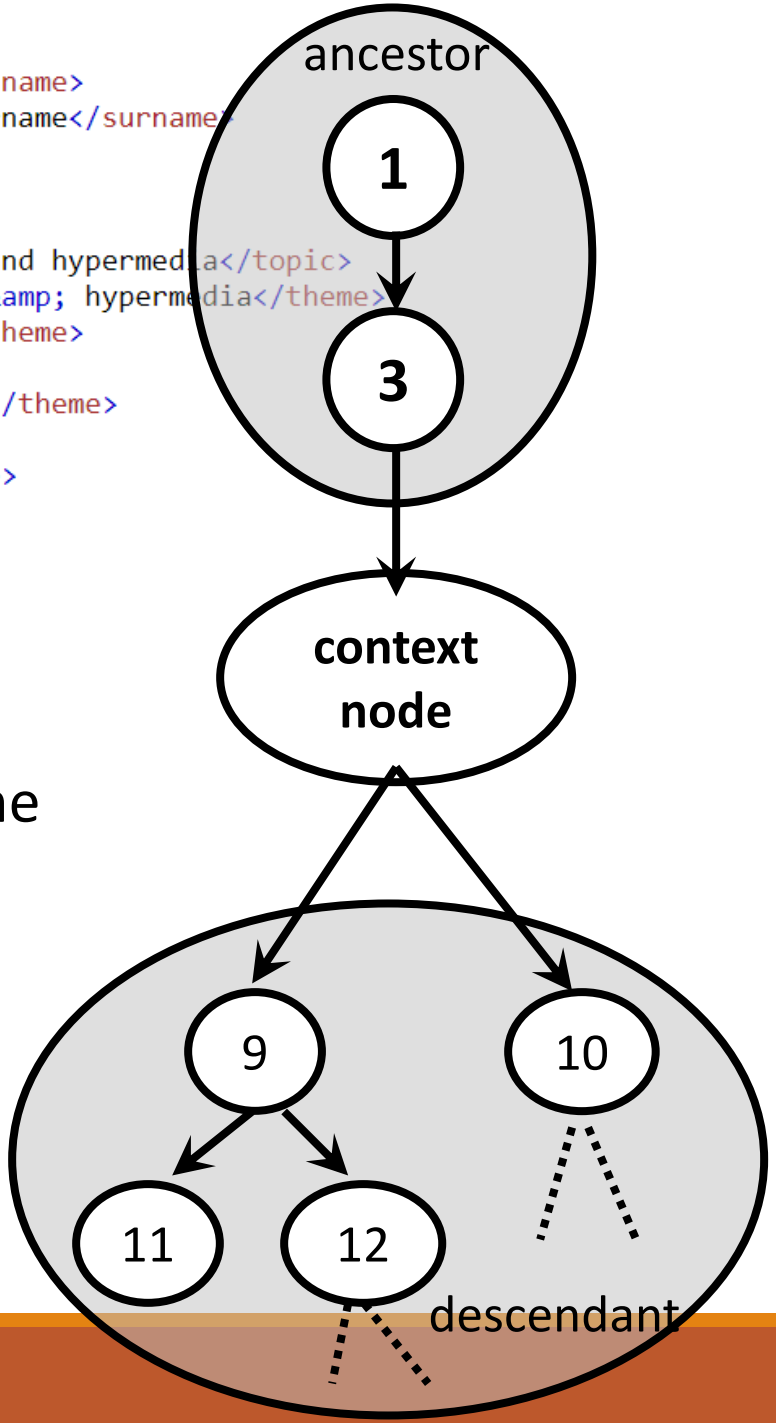
```
<course>
    <author>
        <name>student's name</name>
        <surname>student's surname</surname>
    </author>
    <classes kind="lecture" >
        <component id="1">
            <topic>Hypertext and hypermedia</topic>
            <theme>Hypertext &amp; hypermedia</theme>
            <theme>HTML CSS</theme>
            <theme>XML</theme>
            <theme>XML Schema</theme>
            <theme>DTD</theme>
            <theme>XSLT</theme>
            <score>30</score>
        </component>
    </classes>
```

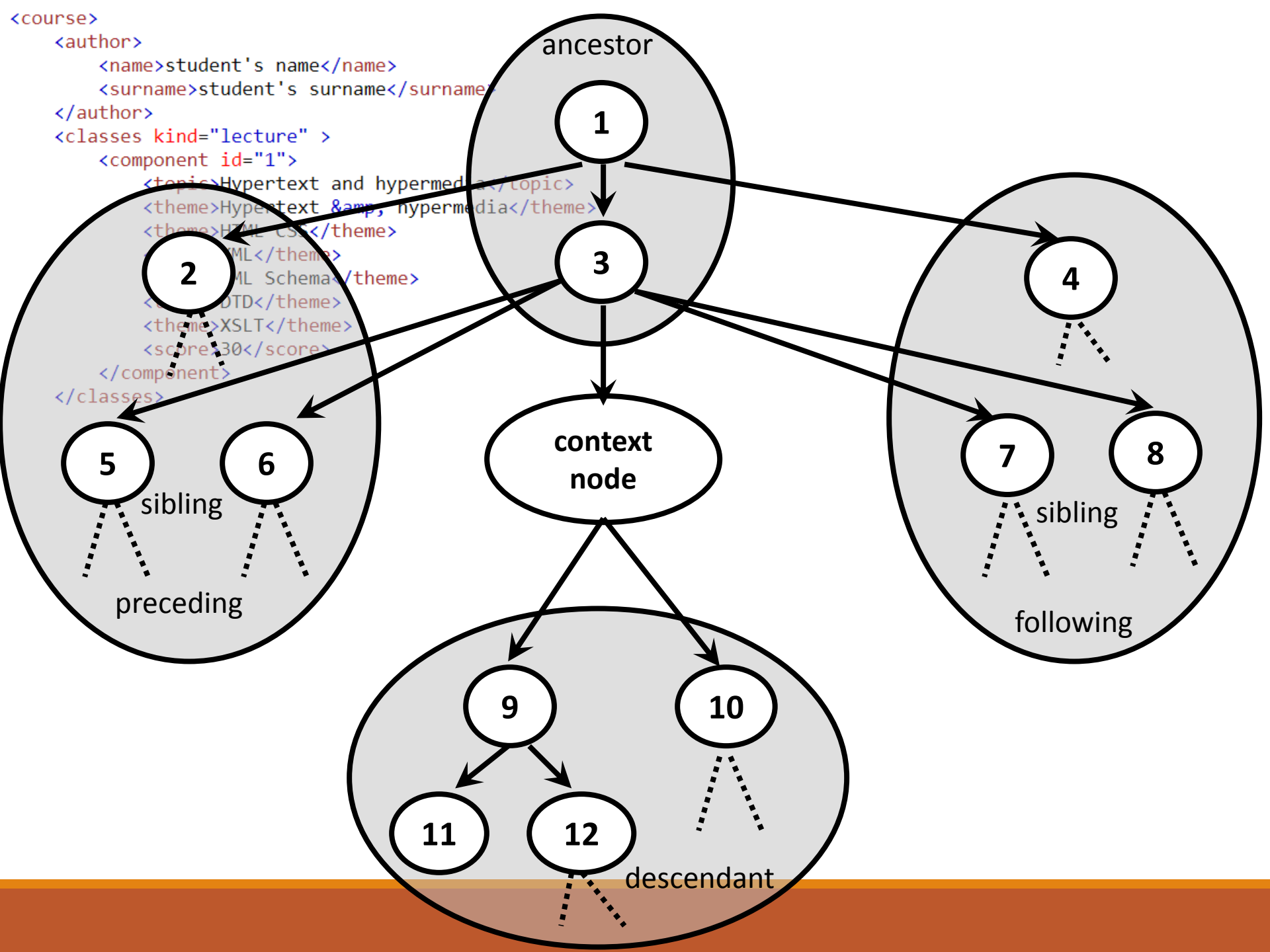


From the context node you can access your parent and ancestors.
'..' represents the parent.
You can go back many levels: ../../classes

```xml
<course>
    <author>
        <name>student's name</name>
        <surname>student's surname</surname>
    </author>
    <classes kind="lecture" >
        <component id="1">
            <topic>Hypertext and hypermedia</topic>
            <theme>Hypertext &amp;amp; hypermedia</theme>
            <theme>HTML CSS</theme>
            <theme>XML</theme>
            <theme>XML Schema</theme>
            <theme>DTD</theme>
            <theme>XSLT</theme>
            <score>30</score>
        </component>
    </classes>
```

component/theme




<course>
<author>
<name>student's name</name>
<surname>student's surname</surname>
</author>
<classes kind="lecture" >
<component id="1">
<topic>Hypertext and hypermedia</topic>
<theme>Hypertext & hypermedia</theme>
<theme>HTML CSS</theme>
<theme>XML</theme>
<theme>XML Schema</theme>
<theme>DTD</theme>
<theme>XSLT</theme>
<score>30</score>
</component>
</classes>
ancestor
1
2
3
4
5
6
sibling
preceding
context node
7
8
sibling
following
9
10
11
12
descendant

Axis

self
child
descendant
parent
ancestor
following-sibling
preceding-sibling
following
preceding
attribute
namespace
descendant-or-self
ancestor-or-self

Node test
by kind of node
node()
text()
comment()
processing-instruction()
attribute()
by name

.//classes/@kind
expands to
self::**node()**/
descendant-or-self::**node()**/ **child**::classes/
attribute::kind

Some built-in XPath functions

Text
- concat(s1, s2, ...) substring(s, pos, len)
  starts-with(s1, s2) contains(s1, s2) string-length(s) translate(s, t1, t2)

Numbers
- floor(x) ceiling(x) round(x)

Nodes
- name(n?) local-name(n?) namespace-uri(n?)

Sequences
- count(S) sum(S) min(S) max(S) avg(S) empty(S) reverse(S) distinct-values(S)

Context
- current() position() last()

Operators

Arithmetic

+ - * div mod

Logical values

and or

Comparison operators

= != < <= > >=

Predicates

additional logical conditions for filtering

/path[predicate]

- they allow you to check properties that can not be expressed in the nodes themselves
- any XPath expression
- predicates may contain functions and operators
- only nodes for which the predicate is true are included in the result

```
<course>
    <author>
        <name>student's name</name>
        <surname>student's surname</surname>
    </author>
    <classes kind="lecture" >
        <component id="1">
            <topic>Hypertext and hypermedia</topic>
            <theme>Hypertext &amp;amp; hypermedia</theme>
            <theme>HTML CSS</theme>
            <theme>XML</theme>
            <theme>XML Schema</theme>
            <theme>DTD</theme>
            <theme>XSLT</theme>
            <score>30</score>
        </component>
    </classes>
```

component/theme[last()]
component/theme[position()<3]
/course/author/name
author | classes/component/topic
theme[3]
classes[@kind='lecture']

XSL (*eXtensible Stylesheet Language*)

SGML(1986) ($\rightarrow$ XML)

DSSSL ($\rightarrow$XSL)

- *Document Style and Semantics Specification Language*
- language for processing and transforming SGML documents into a form that can be displayed or printed
- ***A HUGE MONSTER OF A LANGUAGE***

XSL (*eXtensible Stylesheet Language*)

XSLT *(XSL Transformation)*

XSL FO *(XSL Formating Objects)*

A language that allows you to transform and display data from XML documents
What would have happened if there hadn't been an XSLT

XML document without XSL

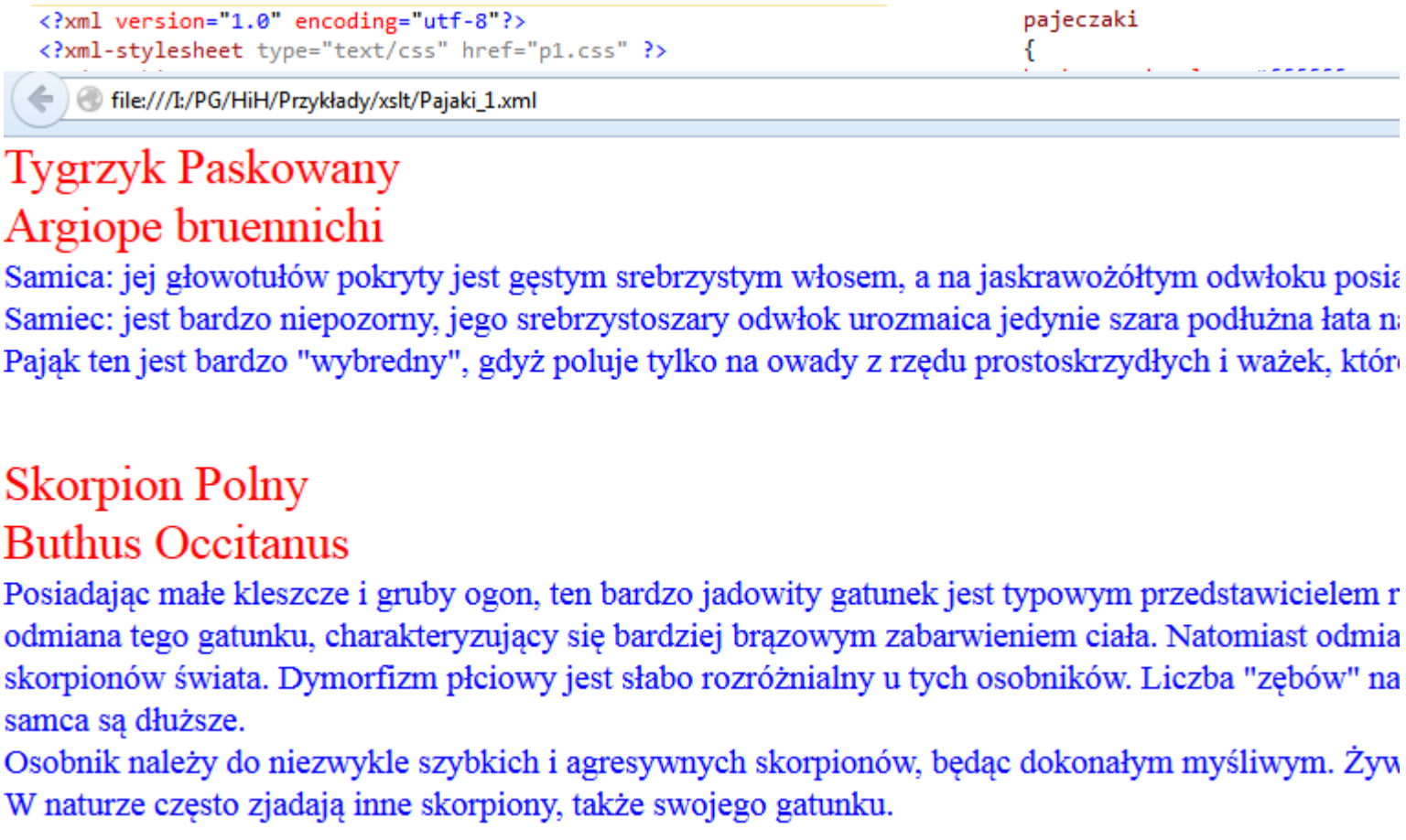
XML document with XSL

Displaying XML using CSS

```xml
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/css" href="p1.css" ?>
<pajeczaki>
  <pajaki gatunek="tygrzyk"  chroniony="tak" wyst_polska="tak">
    <nazwa jezyk="polska">Tygrzyk Paskowany</nazwa>
    <nazwa jezyk="łacińska">Argiope bruennichi</nazwa>
    <opis czego_dotyczy="Cechy">
      Samica: jej głowotułów pokryty jest gęstym srebrzystym włose
    </opis>
    <opis czego_dotyczy="Pokarm">
      Pająk ten jest bardzo "wybredny", gdyż poluje tylko na owady
    </opis>
  </pajaki>
  <skorpiony gatunek="polny" chroniony="tak" wyst_polska="nie">
    <nazwa jezyk="polska">Skorpion Polny</nazwa>
    <nazwa jezyk="lacińska">Buthus Occitanus</nazwa>
    <opis czego_dotyczy="Cechy">
      Posiadając małe kleszcze i gruby ogon, ten bardzo jadowity g
    </opis>
    <opis czego_dotyczy="Pokarm">
      Osobnik należy do niezwykle szybkich i agresywnych skorpionó
    </opis>
  </skorpiony>
</pajeczaki>
```

```css
pajeczaki
{
background-color: #ffffff;
width: 100%;
}
pajaki, skorpiony
{
display: block;
margin-bottom: 30pt;
margin-left: 0;
}
nazwa
{
display: block;
color: #FF0000;
font-size: 20pt;
}
opis
{
display: block;
color: #0000FF;
font-size: 14pt;
}
```

Displaying XML using CSS



XSLT

Extensible Stylesheet Language Transformations

CSS – Cascading Style Sheet

- style sheet determines the style or the appearance of tags (HTML or XML)
- we can define fonts, margins, colours, borders, …

XSLT stylesheet

- a document that generates data based on XML document, the resulting document may or may not contain formatting information
- a complete high-level language for manipulating an XML documents
- it does not replace existing programming languages but complements them

Advantages

- Independent of programming. Transformations are written in a separate XSL file which is an XML document.
- Output can be changed by simply modifying the transformations in an XSL file. No need to change any code. So Web designers can edit the stylesheet and can see the change in the output quickly.

XSLT

Extensible Stylesheet Language Transformations

the language of transforming XML trees

the basic processing paradigm is pattern matching

declarative language

data-driven language (push processing model)

- the code is executed in response to the data;
- the code is executed nondeterministically

XSLT

importance XSLT

- allows you to work with XML documents
  - we do not have to write programs to process XML
- different document type from XML (HTML, …)
- the same transformation can be applied to many XML documents
- different XSLT can be applied to the same XML document

Example

```
<?xml version="1.0" encoding="iso-8859-2"?>
<!--Temat: Drzewo Genealogiczne (XML + XSD)-->
<?xml-stylesheet type="text/xsl" href="bk-drzewo.xsl"?
<drzewo xmlns:xsi="http://www.w3.org/2001/XMLSchema-in
        <osoba plec="kobieta">
                <imie>Halina</imie>
                <nazwisko>Abecka</nazwisko>
                <rodowe>XXXX</rodowe>
                <data>1940-08-14</data>
                <miejsce>Reda</miejsce>
                <informacje>Babcia Halina przez wieksz
                <partner>
                        <osoba plec="mezczyzna">
                                <imie>wiktor</imie>
                                <nazwisko>Abecki</nazw
                                <data>1937-07-03</data
                                <miejsce>Zakopane</mie
                                <informacje>Dziadka wi
                        </osoba>
                </partner>
                <dzieci>
                                                obieta">
                                                Ewa</imie>
                                                sko>Babecka</naz
                                                e>Abecka</rodowe
                                                1961-10-10</data
                                                ce>Ostroda</miej
                                                macje>Moja mama,
                                                er>
```

Halina Abecka

Data urodzenia: 1940-08-14
Miejsce urodzenia: Reda
Informacje: Babcia Halina przez wiekszosc czasu odpoczywa.

Ewa Babecka

Data urodzenia: 1961-10-10
Miejsce urodzenia: Wejherowo
Informacje: Moja mama, dzwoni do mnie raz w tygodniu.

Jola Cebecka

Data urodzenia: 1964-09-21
Miejsce urodzenia: Wejherowo
Informacje: Ciocia Jola to najbardziej wykrecona osoba z rodziny.

Katarzyna Ebecka

Data urodzenia: 1962-07-07
Miejsce urodzenia: Wejherowo
Informacje: Ciocia Kasia mieszka niedaleko ode mnie i czesto do nas wpada.

Halina Abecka

Nazwisko rodowe: XXXX
Data urodzenia: 1940-08-14
Miejsce urodzenia: Reda
Informacje: Babcia Halina przez wiekszosc czasu odpoczywa.

Wiktor Abecki

Data urodzenia: 1937-07-03
Miejsce urodzenia: Zakopane
Informacje: Dziadka Wiktora juz nie ma na tym swiecie.

Ewa Babecka

Nazwisko rodowe: Abecka
Data urodzenia: 1961-10-10
Miejsce urodzenia: Ostroda
Informacje: Moja mama, dzwoni do mnie raz w tygodniu.

Edward Babecki

Data urodzenia: 1960-03-15
Miejsce urodzenia: Gdynia
Informacje: Moj Tata-sponsor, wymienia ze mna jakies 30 pelnych zdan w ciagu roku.

Jan Szymon Babecki

Data urodzenia: 1982-02-15
Miejsce urodzenia: Ostroda
Informacje:

Maciej Mikolaj Babecki

Data urodzenia: 1984-10-03
Miejsce urodzenia: Ostroda
Informacje: Brachol Maciek to zapalony hiphopowy DJ, w wolnym czasie studiuje na PG.

Halina Abecka

Nazwisko rodowe: XXXX
Data urodzenia: 1940-08-14
Miejsce urodzenia: Reda
Informacje: Babcia Halina przez wiekszosc czasu odpoczywa.

Wiktor Abecki

Data urodzenia: 1937-07-03
Miejsce urodzenia: Zakopane
Informacje: Dziadka Wiktora juz nie ma na tym swiecie.

Ewa Babecka

Nazwisko rodowe: Abecka
Data urodzenia: 1961-10-10
Miejsce urodzenia: Ostroda
Informacje: Moja mama, dzwoni do mnie raz w tygodniu.

Edward Babecki

Data urodzenia: 1960-03-15
Miejsce urodzenia: Gdynia
Informacje: Moj Tata-sponsor, wymienia ze mna jakies 30 pelnych zdan w ciagu roku.

Jan Szymon Babecki

Data urodzenia: 1982-02-15
Miejsce urodzenia: Ostroda
Informacje:

Maciej Mikolaj Babecki

Data urodzenia: 1984-10-03
Miejsce urodzenia: Ostroda
Informacje: Brachol Maciek to zapalony hiphopowy DJ, w wolnym czasie studiuje na PG.

Anna Nowak

Nazwisko rodowe: XXXX
Data urodzenia: 1940-08-14
Miejsce urodzenia: Reda
Informacje: Babcia

Andrzej Nowak

Data urodzenia: 1940-07-03
Miejsce urodzenia: Torun
Informacje: Dziadek

Katarzyna Lusnia

Nazwisko rodowe: Nowak
Data urodzenia: 1963-10-10
Miejsce urodzenia: Kalisz
Informacje: Mama

Jan Lusnia

Data urodzenia: 1960-03-15
Miejsce urodzenia: Gdynia
Informacje: Tata

Edward Szymon Lusnia

Data urodzenia: 1982-02-15

XSLT

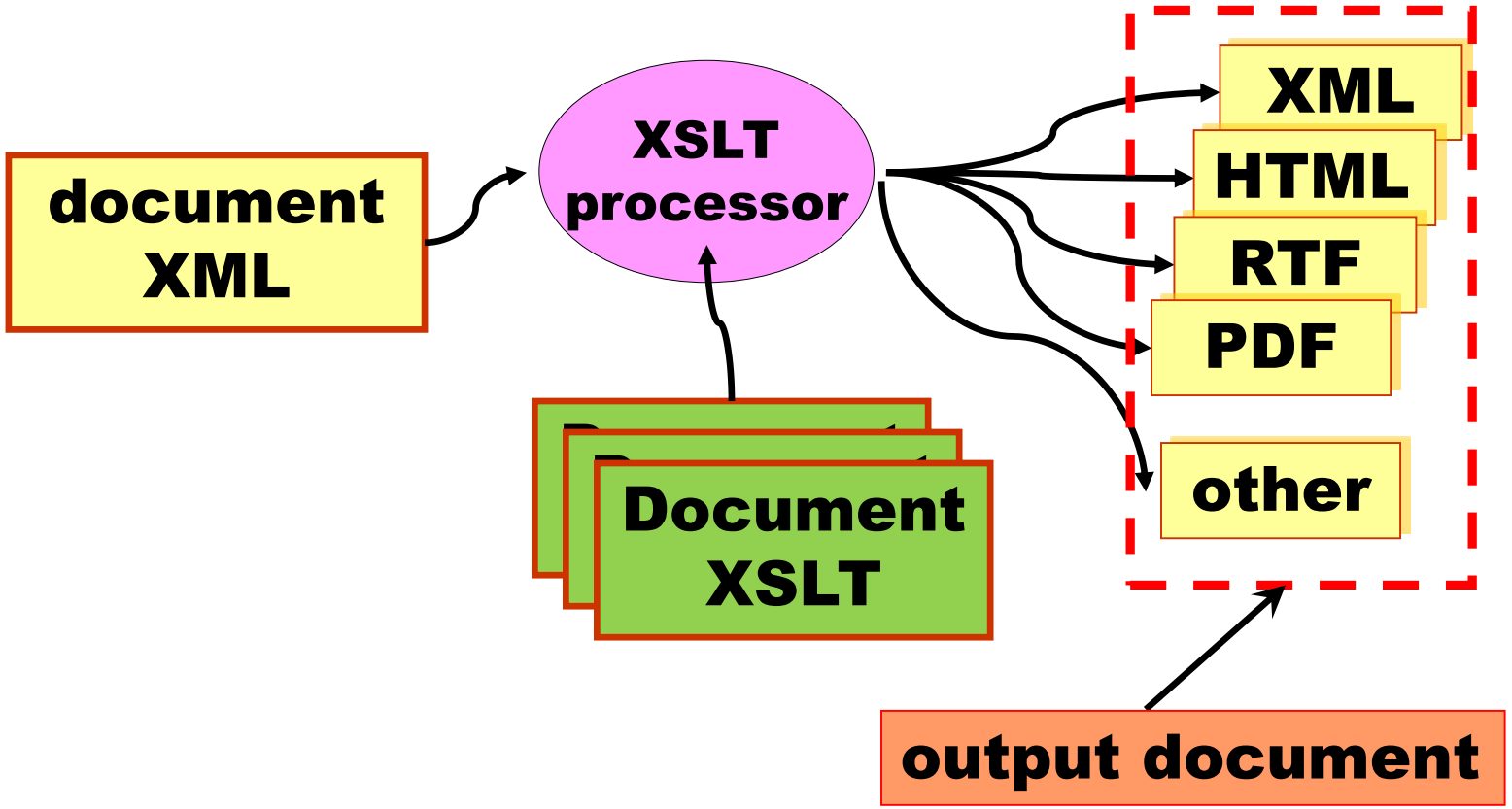
XML

```
<?xml version="1.0" encoding="iso-8859-2"?>
<!--Temat: Drzewo Genealogiczne (XML + XSD)-->
<?xml-stylesheet type="text/xsl" href="bk-drzewo.xsl"?>
<drzewo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="bk-drzewo.xsd">
        <osoba plec="kobieta">
                <imie>Halina</imie>
                <nazwisko>Abecka</nazwisko>
                <rodowe>XXXX</rodowe>
                <data>1940-08-14</data>
                <miejsce>Reda</miejsce>
                <informacje>Babcia Halina przez wiekszosc czasu odpoczywa.</informacje>
                <partner>
                        <osoba plec="mezczyzna">
                                <imie>wiktor</imie>
                                <nazwisko>Abecki</nazwisko>
                                <data>1937-07-03</data>
                                <miejsce>Zakopane</miejsce>
                                <informacje>Dziadka wiktora juz nie ma na tym swiecie.</informacje>
                        </osoba>
                </partner>
                <dzieci>
                        <osoba plec="kobieta">
                                <imie>Ewa</imie>
                                <nazwisko>Babecka</nazwisko>
                                <rodowe>Abecka</rodowe>
                                <data>1961-10-10</data>
                                <miejsce>Ostroda</miejsce>
                                <informacje>Moja mama, dzwoni do mnie raz w tygodniu.</informacje>
                                <partner>
```

```
<?xml version="1.0" encoding="iso-8859-2"?>
<!--Temat: Drzewo Genealogiczne (XML + XSD)-->
<?xml-stylesheet type="text/xsl" href="bk-drzewo.xsl"?>
<drzewo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="bk-drzewo.xsd">
        <osoba plec="kobieta">
                <imie>Anna</imie>
                <nazwisko>Nowak</nazwisko>
                <rodowe>XXXX</rodowe>
                <data>1940-08-14</data>
                <miejsce>Reda</miejsce>
                <informacje>Babcia</informacje>
                <partner>
                        <osoba plec="mezczyzna">
                                <imie>Andrzej</imie>
                                <nazwisko>Nowak</nazwisko>
                                <data>1940-07-03</data>
                                <miejsce>Torun</miejsce>
                                <informacje>Dziadek</informacje>
                        </osoba>
                </partner>
                <dzieci>
                        <osoba plec="kobieta">
                                <imie>Katarzyna</imie>
                                <nazwisko>Lusnia</nazwisko>
                                <rodowe>Nowak</rodowe>
                                <data>1963-10-10</data>
                                <miejsce>Kalisz</miejsce>
                                <informacje>Mama</informacje>
                                <partner>
                                        <osoba plec="mezczyzna">
                                                <imie>Jan</imie>
                                                <nazwisko>Lusnia</nazwisko>
                                                <data>1960-03-15</data>
                                                <miejsce>Gdynia</miejsce>
                                                <informacje>Tata</informacje>
                                        </osoba>
                                </partner>
                                <dzieci>
                                        <osoba plec="mezczyzna">
                                                <imie>Edward</imie>
                                                <drugie>Szymon</drugie>
                                                <nazwisko>Lusnia</nazwisko>
                                                <data>1982-02-15</data>
                                                <miejsce>Ostroda</miejsce>
                                                <informacje></informacje>
                                        </osoba>
                                        <osoba plec="kobieta">
                                                <imie>Hanna</imie>

                                                <nazwisko>Lusnia</nazwisko>
                                                <data>1995-05-20</data>
                                                <miejsce>Ostroda</miejsce>
                                                <informacje>siostra</informacje>
                                        </osoba>
```

XSLT


document
XML
XSLT
processor
Document
XSLT
XML
HTML
RTF
PDF
other
output document

XSLT

From a single XML file, we are able to generate:

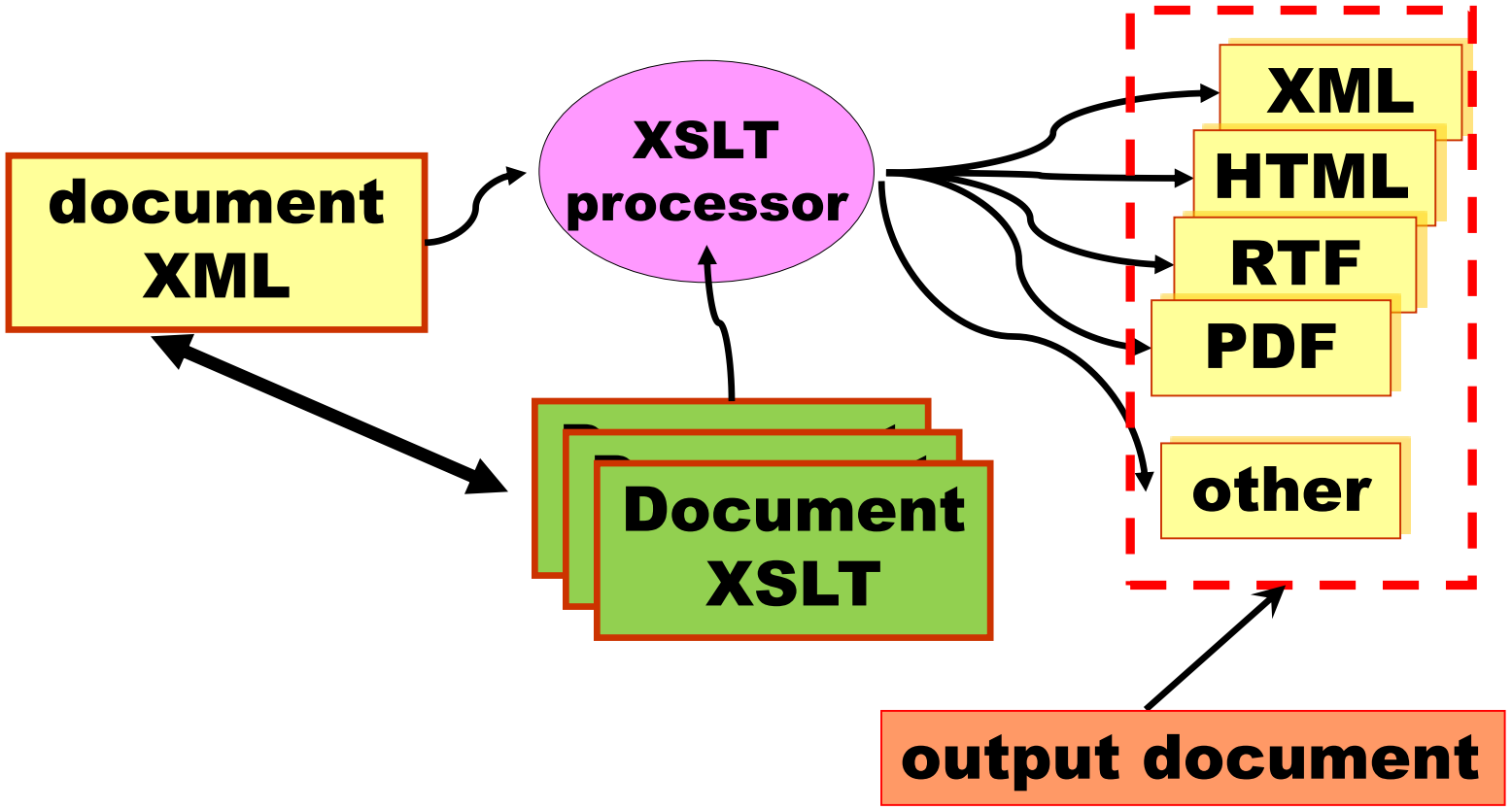
- an HTML webpage, to publish the data on the web
- a CSV file, to easily import into a database/spreadsheet
- a PDF file for printing
- another XML file, with different tag names or layout
- other text files

XSLT

XSLT processors

- XT (*James Clark*)
  - One of the first
  - Created in Javie
- Xalan (*Apache*)
  - Easy to use in Java and C++ environments
- Msxml (*Microsoft*)
  - IE
- SAXON
- LotusXSL, Koala XSL Engine, iXSLT, XT, …

XSLT


document
XML
XSLT
processor
XML
HTML
RTF
PDF
other
Document
XSLT
output document

XSLT

command line (eg Xalan)

- java org.apache.xalan.xslt.Process -IN file.xml -XSL file.xsl -OUT file.html

in XML file

- <?xml-stylesheet type= ” text/xsl” href= ” sheet.xsl”?>

Stylesheet processing

XSLT processor get a document and a stylesheet

It starts a (breadth-first) traversal at the **root node** and checks to see if there is a template match

- If so, it applies the template and looks for an “xsl:apply-templates” element
  - If such an element exists, it continues the traversal
  - if no such element exists, the traversal stops
- If not, it traverses down the tree looking for a template match with some other node of the tree

Structure of the XSLT document

XML declaration

xsl:stylesheet tag

- xmlns:xsl attribute: URI for XSLT namespace
  - xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
- version

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
…templates and transformation rules go here …
</xsl:stylesheet>
```

Structure of the XSLT document

xsl:attribute-set	it defines a set of attributes
xsl:decimal-format	defines how the numbers are displayed
xsl:import	import style sheets
xsl:include	include style sheets
xsl:key	key
xsl:namespace-alias	indicates the namespace in the resulting document
xsl:output	determines the type of result document
xsl:param	allows you to create parameters
xsl:preserve-space	retains white characters in the specified element
xsl:strip-space	removes whitespace from the specified element
xsl:template	defines the template
xsl:variable	defines variables

Structure of the XSLT document

xsl:apply-imports

xsl: apply-templates

xsl: attribute

xsl:call-template

xsl:choose

xsl:comment

xsl:copy

xsl:copy-of

xsl:element

xsl:fallback

xsl:for-each

xsl:if

xsl:message

xsl: number

xsl: otherwise

xsl:processing-instruction

xsl:sort

xsl:text

xsl:value-of

xsl:when

xsl:with-param

Structure of the XSLT document

hello.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/T
ransform"  versi
  <xsl:output me
  <xsl:template match="/">
    <b><xsl:value-of
              select="hi"/></b>
  </xsl:template>
</xsl:stylesheet>
```

hello.xml

```
<?xml version="1.0"?>
<hi>Hello XSLT!</hi>
```

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

Hello XSLT!

The format of the output document

xsl:output

attributes

- *method* defines the output format
  - html, xml, text
- *encoding*, *version*, ...

default method: xml

XSLT extraction of information

xsl:value-of

- ***select***
  - determines the node from which we want to get the information
  - the attribute contains an XPath expression or XSLT function

xsl:value-of - only **first node** from set

```
<xsl:value-of select="node"/>
```

```
<?xml version="1.0" encoding="iso-8
- <ptaki xmlns:xsi="http://www.w3.
  - <ptak numer="1">
    - <nazwa>
        <polska>Zimorodek</polska>
        <łacińska>Alcedo atthis</łac
      </nazwa>
      <gromada>ptaki</gromada>
    - <występowanie>
        <region_swiata>Południowa
          Malajski.</region_swiata>
        <środowisko_życia>lądowe</
        <Polska>tak</Polska>
      </występowanie>
    - <opis>
        <cechy_charakterystyczne>B
          połyskiem,środek grzbietu i sterówki błękitne, gardło i policzki białe, ciemne lotki, niebiesko-zielone ze
          chorągiewki i cynamonowy spód ciała. Ma nieproporcjonalnie długi i mocny dziób oraz bardzo krótkie n
          Lata szybkim, prostoliniowym lotem, zwykle nisko nad wodą, odzywając się wysokim przenikliwym
          głosem.</cechy_charakterystyczne>
        <pożywienie>Drobne rybki, które zimorodek zdobywa
      - <wymiary>
          <długość_ciała>17cm</długość_ciała>
          <rozpiętość_skrzydeł>28cm</rozpiętość_skrzydeł>
          <waga>56g</waga>
        </wymiary>
      </opis>
      <chroniony>tak</chroniony>
    - <galeria>
        <zdjęcie numer="1">img/zimorodek_1.jpg</zdjęcie>
        <zdjęcie numer="2">img/zimorodek_2.jpg</zdjęcie>
        <zdjęcie numer="3">img/zimorodek_3.jpg</zdjęcie>
      </galeria>
```


```
<h3>Zdjęcia</h3>
<div id="zdjęcia">
   <xsl:for-each select="galeria/zdjęcie">
      <img>
         <xsl:attribute name="src">
            <xsl:value-of select="."/>
         </xsl:attribute>
      </img>
   </xsl:for-each>
</div>
<p>
   <br/>
   <h4>
      <xsl:value-of select="data_modyfikacji"/>
   </h4>
</p>
<br/>
</xsl:template>
```

XSLT - templates

xsl:template

A template contains rules to apply when a specified node is matched.

The template can be called by matching the node of the XML document tree to the *match* attribute

Nodes are matched and not selected, they are processed as they are encountered by the XML processor

XSLT - templates

xsl:template
- match
- mode
- name

```
<?xml version="1.0"?>
<R>
    <A>1</A>
    <B>2</B>
    <A>3</A>
</R>
```

```
<xsl:template match="R">
  …
</xsl:template>

<xsl:template match="A">
  …
</xsl:template>

<xsl:template match="B">
  …
</xsl:template>
```

```
<xsl:template match= "XPath-expression">
  …
</xsl:template>
```

ogólny format szablonu

XSLT - templates

xsl:template

- match
- mode
- name

using templates

- xsl:apply-templates
- xsl:call-template

templates must not be nested

XSLT - templates

the template element must have one of the attributes:

- **match**
  - defines the transformation for the nodes described by this attribute

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

- **name**
  - gives the template a name

<xsl:template name="...">...</xsl:template>

XSLT - templates

template with match attribute

- template definition:

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

- using template:
  - all of the children of the current node

```
<xsl:apply-templates/>
```

  - selecting the nodes to which the template is used

```
<xsl:apply-templates select="..." />
```

```
<?xml version="1.0"?>
<R>
        <A>1</A>
        <B>2</B>
        <A>3</A>
</R>
```

XSLT - templates

template with name attribute

- template definition:

```
<xsl:template name="dosomething">...</xsl:template>
```

- using template :

```
<xsl:call-template name=" dosomething "/>
```

looks like a traditional programming language

we have full control over the way of execution the code

XSLT - templates

```
...
<xsl:template match="mammals">
<h1><xsl:apply-templates/></h1>
</xsl:template>
...
<xsl:template match="birds">
<h1><xsl:apply-templates/></h1>
</xsl:template>
…
```

```
...
<xsl:template match="mammals | birds">
<h1><xsl:apply-templates/></h1>
</xsl:template>
...
```

XSLT - templates

If no rule matches the node?

built-in template

- for document root and elements:
  - apply templates to children
- for text nodes and attributes:
  - copy text value to result
- for comments and processing instructions
  - do nothing

XSLT - templates

template with mode

- possibility to process the same set of nodes many times

XSLT - templates

```
<xsl:template match="root">
    <xsl:apply-templates mode="tryb1"/>
    <xsl:apply-templates mode="tryb2"/>
</xsl:template>

<xsl:template match="elem" mode="tryb1">
    <xsl:text>tryb1:</xsl:text>
    <xsl:value-of select="."/>
</xsl:template>

<xsl:template match="elem" mode="tryb2">
    <xsl:text>tryb2:</xsl:text>
    <xsl:value-of select="."/>
</xsl:template>
```

Resolving matching rules conflicts

Which template will be used?

there is no template for the given context
- built-in template will be used

there is one template for the given context
- this template will be used

there are 2 templates for the given context
- a more specific template will be applied

there are 2 templates for the given context, both equally specific
- a template with higher priority will be applied

there are 2 templates: the same context, priority, both equally specific
- a template that appears later will be applied

XSLT

The XSLT language allows you to

- conditional processing,
- looping
- parameterisation
  - data from outside the xml and xslt file can affect the result

XSLT - conditional processing

xsl:if

- **test**

no else

```
<xsl:if   test="gender='woman‘ ">
            <font class="napis">Wife</font>
</xsl:if>
<xsl:if   test="gender='man‘ ">
            <font class="napis">Husband</font>
</xsl:if>
```

```
<xsl:if test="$srodowisko='wodne'">
        <xsl:attribute name="style">
                background:url(img/tab/morze.jpg) center no-repeat;
        </xsl:attribute>
        wodne
</xsl:if>
<xsl:if test="$srodowisko='ladowe'">
        <xsl:attribute name="style">
                background:url(img/tab/ziemia.jpg) center no-repeat;
        </xsl:attribute>
        l±dowe
</xsl:if>
```

iary masa	Gromada	Czy lata?	Środowisko życia	Region
0.3kg	ptak	nie	lądowe	Różnie
20kg	ptak	nie	lądowe	Mauritius
250kg	ptak	nie	wodne	Nowa Zelandia

XSLT

xsl:choose : conditional processing

- xsl:when : determines the condition
  - test
- xsl:otherwise : optional, final instructions, used if no condition satisfied
- only first branch with satisfied condition processed.

```
<xsl:choose>
        <xsl:when test="@gender='woman‘ ">
                <img src="bk-female.png" alt="woman"/>
        </xsl:when>
        <xsl:when test="@ gender='man‘ ">
                <img src="bk-male.png" alt="man"/>
        </xsl:when>
</xsl:choose>
```

XSLT - conditional processing

```
<xsl:choose>
        <xsl:when test="position()=first()">
                Do something for last element
        </xsl:when>
        <xsl:when test="position()=last()">
                Do something for last element
        </xsl:when>
        <xsl:otherwise>
                Do something for other elements
        </xsl:otherwise>
</xsl:choose>
```

XSLT - loop

xsl:for-each

- **select**

the XSLT processor processes all nodes corresponding to the pattern given in the *select* attribute

possibility of sorting - xsl:sort

```
<xsl:for-each select="XPath expression">
        some instructions
</xsl:for-each>
```

Sorting

xsl:sort

attributes

- **select**
  - sorting by element or attribute
- **order**
  - *ascending*, *descending*
- **case-order**
  - specifies letter size priority, *upper-first* , *lower-first*
- **lang**
- **data-type**
  - sorting letters or numbers (*text*, *number*)

possibility of multiple sorting criteria

Sorting

xsl:sort

- xsl:for-each
- xsl:apply-templates

xsl:sort the first instruction in for-each

```
<xsl:apply-templates >
        <xsl:sort select="wezel"/>
</xsl:apply-templates>
```

```
<xsl:for-each select="osoba">
  <xsl:sort select="@plec"/>
  <xsl:sort select="nazwisko"/>
  <xsl:sort select="imie"/>
    <center>
    <xsl:choose>
      <xsl:when test="@plec='kobieta'">
            <table class="kobieta" align="center"> <xsl:call-template name="tabelka"/>
            </table>
      </xsl:when>
      <xsl:when test="@plec='mezczyzna'">
            <table class="mezczyzna" align="center"> <xsl:call-template name="tabelka"/>
            </table>
      </xsl:when>
    </xsl:choose>
    </center>
</xsl:for-each>
```

Numbering

xsl:number

it can be located anywhere in the templates also in for-each element

defining the form of the number - attribute *format*

- 1 1,2,3...
- 01 01,02,03,...
- a a,b,c,...z,aa...
- B A,B,C...
- i i,ii,iii,iv,...
- ....

Numbering

level attribute

- = ”any”
  - continuous numbering of elements regardless of their parent element
- = ”multiple”
  - multi-level numbering (eg 1.2; 3.2.4, …)

the possibility of grouping numbering

- grouping-separator , grouping-size
  - grouping-separator=”.” , grouping-size=”3”
  - 12345678 -> 12.345.678

```
<xsl:template match="odsylacze">
<h3>Ciekawe linki:</h3>
        <xsl:for-each select="odsylacz">
                <xsl:number value="position()" format="I"/>
                <a class="obrazek">
                        <xsl:attribute name="href"><xsl:value-of select="@href"/></xsl:attribute
                        <xsl:attribute name="title"><xsl:value-of select="@alt"/></xsl:attribute
                        <xsl:text> </xsl:text><xsl:value-of select="."/><br/>
                </a>
        </xsl:for-each>
</xsl:template>
```

Opis motylka:

Przednie skrzydło czarne z ukośnymi, białożółtymi paskami. Tylne skrzydło pomarańczowoczerwone z czarnymi pl
pomarańczowoczerwony z rzędem drobnych, czarnych plamek na stronie grzbietowej. Motyle latają głównie w cza
żywiąc się naktarem - głównie ostów (Carduus spp.). Przepoczwarczenie następuje w luźnym oprzędzie w ściółce
jesienią na różnych roślinach zielnych takich jak jasnota, pokrzywa. Po przezimowaniu żerują na leszczynie, malini
wiciokrzewie. Samice składają jaja w małych złożach na krawędzi liści roślin żywicielksich.

Ciekawe linki:

I Moths and Butterflies of Europe
II UK Moths
III Schmetterlingsseiten von Jürgen Rodeland
IV Insekten Box

Variables

xsl:variable

definition

- <xsl:variable name="VarName">value</xsl:variable>
- <xsl:variable name="VarName" select ="'’value’"/>

reference to variable

- <xsl:value-of select=”$VarName”/>

Variables

local

```
<xsl:template name="...">
        <xsl:variable name="...">...</xsl:variable>
</xsl:template>
```

global

```
<xsl:stylesheet ...>
        <xsl:variable name="...">...</xsl:variable>
        ...
</xsl: stylesheet >
```

Variables??

Constant (*name given to a certain value*)

their values can not be modified (*read only*)

advantages of variables

- they make reading the code easier
  - complex expression saved as a variable
  - the ability to break complex expressions into parts
- reuse
  - performance enhancement especially for complex expressions that result in a fragment of the tree
- saving the value of nodes that are currently unavailable

Variables

simple

- they contain single values
- usually used to insert the same values in many places in a document

complex

- they contain sets of nodes and tree fragments

```
<xsl:template match="zwierzak">

        <!-- tytuł: -->
        <h2><xsl:value-of select="nazwa/polska"/>, (<i><xsl:value-of select="nazwa/lacinska"/></i>)</h2>
        <img align="left" hspace="10" src="img/archeopteryks/tb/archeopteryks_05.jpg"/>
        <xsl:apply-templates select="wystepowanie"/> <!-- szablon wystepowania -->
        <xsl:apply-templates select="opis"/> <!-- szablon opisu -->

        <xsl:variable name="nazwa_zwierza">     <!-- zmienna nazwy -->
                <xsl:value-of select="nazwa/polska"/>
        </xsl:variable>

        <h4>Zdjęcia:</h4>
                <xsl:for-each select="zdjecia">
                        <img src="{.}" alt="{$nazwa_zwierza}">
                                <xsl:comment><xsl:value-of select="."/></xsl:comment>
                        </img>
                </xsl:for-each>
        <br/>

        <xsl:if test="position()!=last()">      <!-- linie rozdzielaj±ce zwierzaki -->
                <br/><br/><hr/><br/>
```

Opis:

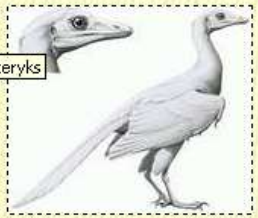
- Wygląd: Duży, masywny ptak o uwstecznionych skrzydłach z charakte
  biały.
- Dodatkowe informacje: Pierwszy raz opisany został przez żeglar
  posiadający masywne, krótkie nogi i uwstecznione skrzydła i ogon.
- Pokarm: Brak dostatecznie wiarygodnych danych. Według przekazu
  że żywił się kamieniami i żelazem (prawdopodobnie dodo połykały ka
- Wymiary:
  - Długość ciała: 75 cm
  - Masa: 20 kg

Zdjęcia:

Opis:

- Wygląd: Wygląd stanowi zagadkę, jak to u zwierzą kopalnych bywa. Wiadomo, że posiadał p
  ptaka.
- Dodatkowe informacje: Gatunek znany jest z sześciu skamielin znalezionych w litografic
  okolicach miejscowości Solnhofen w Bawarii.
- Pokarm: Owady
- Wymiary:
  - Długość ciała: 45 cm
  - Masa: 0.3 kg

Zdjęcia:



```
<xsl:variable name="bgcolor">
  <body>#cccccc</body>
  <table>#ffffff</table>
  <row>#cccccc</row>
  <altrow>#ffffff</altrow>
</xsl:variable>

<xsl:template match="/">
  <html>
  <body bgcolor="{$bgcolor/body}">
    <xsl:apply-templates />
  </body>
  </html>
</xsl:template>

<xsl:template match="cars">
  <table bgcolor="{$bgcolor/table}" width="75%">
    <xsl:for-each select="car">
      <tr>
        <xsl:attribute name="bgcolor">
          <xsl:choose>
            <xsl:when test="position() mod 2 = 0">
              <xsl:value-of select="$bgcolor/altrow" />
            </xsl:when>
            <xsl:when test="position() mod 2 = 1">
              <xsl:value-of select="$bgcolor/row" />
            </xsl:when>
          </xsl:choose>
```

Focus	Ford	2000
Golf	Volkswagen	1999
Camry	Toyota	1999
Civic	Honda	2000
Prizm	Chevrolet	2000

Formatting numbers

Converting numerical values into strings

format-number(number, format_pattern, dec_format)

value to format

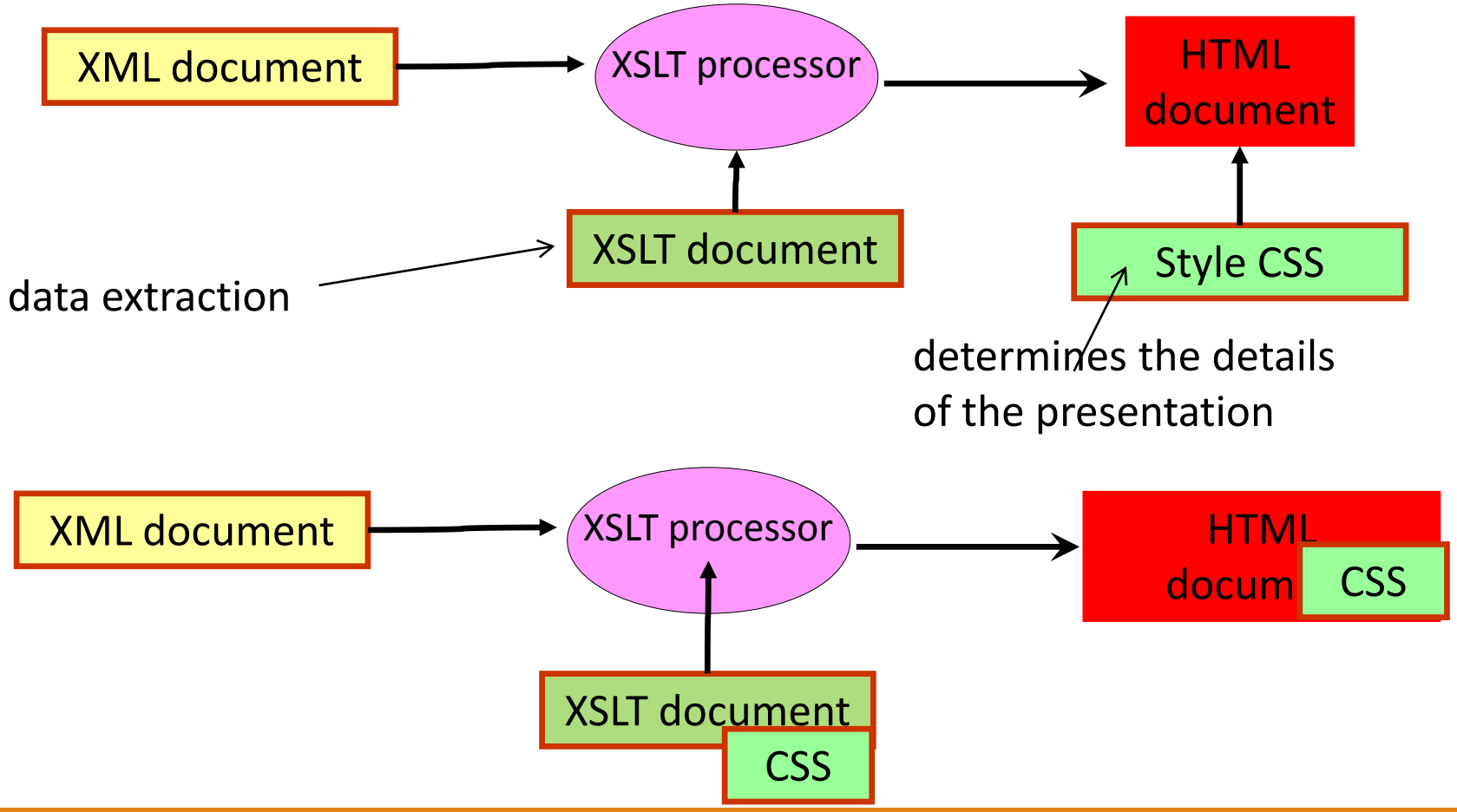
name of the user-defined format

{prefix}number{.fraction}{suffix}

0 Digit
Digit, zero shows as absent
. The position of the decimal point (###.##)
, The group separator for thousands (###,###.##)
% Displays the number as a percentage (##%)

```
<xsl:decimal-format
    name="formatname"
    decimal-separator=""
    grouping-separator=""
    infinity=""
    minus-sign=""
    NaN=""
    percent=""
    per-mille=""
    zero-digit=""
    digit=""
    pattern-separator=""/>
```

XSLT – using CSS



```
body
{
        background-color:cornflowerblue;
}
a
{
        color:indigo;
        text-decoration:underline;
        font-weight:bold;
}
a:link
{
        color:indigo;
        text-decoration:underline;
h1
{
        font-size:xx-large;
        font-family:'Courier New','Arial';
        color:navy;
}
h2
{
        font-size:x-large;
        font-family:'Arial';
        color:purple;
}
h3
{
        font-size:large;
        font-family:'Courier New','Times New Roman';
        color:lime;
}
p
{
        font-size:normal;
        font-family:'Courier New';
}
table
{
        border-style:dotted;
        border-width:2px;
        border-color:red;
}
```


```
<?xml version="1.0" encoding="windows-1250"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns
<xsl:output method="html"/>

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

<xsl:template match="zwierzeta">
        <html>
                <head>
                        <link rel="stylesheet" type="text/css" href="styl2.css"/>
                        <title>Moje zwierzęta</title>
                </head>
                <body>
                        <xsl:for-each select="zwierze">
                                <h1><xsl:value-of select="nazwa_lacinska"/></h1>
                                <h2>(<xsl:value-of select="gatunek"/>)</h2>
```

Panstwo	Region		Miasto		Nocleg	
Filipiny	Nazwa:	*Filezja*	Nazwa:	*Filipia*	Rodzaj:	*Pole namiotowe*
	Opis	*Wyspa jednym malym miastem:)*	Wielkosc:	*<1000m2*	Ulica:	*Dolnk*
	Charakter	*Morski*	Mieszkancy:	*<1000*	Cena	*11.40 USD*
	www	http://www.filezja.com				
Nowa gwinea	Nazwa:	*Nowagwineandia*	Nazwa:	*Gwineanka*	Rodzaj:	*Pole namiotowe*
	Opis	*Śliczny region z jednym malym miastem i mnostwem jezior:)*	Wielkosc:	*<1000m2*	Ulica:	*Swinnkowa*
	Charakter	*Pojezierze*	Mieszkancy:	*<1000*	Cena	*21.20 USD*
	www	http://www.gwineandia.com				
Boliwia	Nazwa:	*Boliwiandia*	Nazwa:	*Cesiowo*	Rodzaj:	*Kwatera prywatna*
	Opis	*Śliczny gorski region z jednym duzym miastem:)*	Wielkosc:	*<1000m2*	Ulica:	*Repete*
	Charakter	*Wiejski*	Mieszkancy:	*1000 - 5000*	Cena	*31.20 USD*
	www	http://www.boliwiandia.com				
Maleary	Nazwa:	*Malearandia*	Nazwa:	*Male*	Rodzaj:	*Hotel*
	Opis	*Śliczny nizinny region z wielkim miastem:)*	Wielkosc:	*>10000m2*	Ulica:	*Trutka*
	Charakter	*Pasterski*	Mieszkancy:	*>10000*	Cena	*61.20 USD*
	www	http://www.malearandia.com				

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:output method="html"/>
    <xsl:strip-space elements="*"/>
    <xsl:template match="/">
        <html>
            <head>
                <title>Miejsca wakacyjne</title>
            </head>

                <style>
                        BODY {SCROLLBAR-FACE-COLOR: #64809B; SCROLLBAR-HIGHLIGHT-COLOR: #64809B; SCROLLBAR-3DLIGHT-COLOR: #000000; SCROLLBAR-ARROW-CO
 #FFFFFF; SCROLLBAR-TRACK-COLOR: #778CA4; SCROLLBAR-DARKSHADOW-COLOR: #000000; }.td_dddddCopy2 { background: #FFD891; border: 1px solid; border-colo
#FFFFFF #7A7A7A #7A7A7A #FFFFFF; font: 10px Verdana, Arial, Helvetica, sans-serif; color: #000000 }
                        .menu_link { font: normal 10px Verdana, Arial, Helvetica, sans-serif; color: #000000; text-decoration: none }
                        .menu_link:hover { font: 10px Verdana, Arial, Helvetica, sans-serif; color: #FFFFFF; text-decoration: underline }
                        .table_border { border: 1px solid; border-collapse: separate; border-color: #FFCC34 #A7B4C5 #FFCC34 #FFCC34 }
                        .table_border2 {border-collapse: separate; border-color: ##000000 ##A7B4C5 ##FFCC34 ##FFCC34 }
                        .border { border-right-style: solid; border-right-width: 1; border-right-color: #FFCC34 }
                </style>

            <body>
                <center>
                <table border="1">
                    <tr>
                        <xsl:attribute name="style">background-color: red; color: black; font-weight: bold; font-family: Lucida;</xsl:attribute>
                        <td><b><i><center><big>Panstwo</big></center></i></b></td>
                        <td colspan="2"><b><i><center><big>Region</big></center></i></b></td>
                        <td colspan="2"><b><i><center><big>Miasto</big></center></i></b></td>
                        <td colspan="2"><b><i><center><big>Nocleg</big></center></i></b></td>
                    </tr>

                    <xsl:for-each select="wakacje/etap">
                        <xsl:sort select="nocleg/cena"/>

                        <tr>
                            <xsl:attribute name="style">background-color: #FFFD8F; font-weight: bold; font-family: Lucida</xsl:attribute>

                            <td rowspan="4"><xsl:value-of select="panstwo"/></td>
```

Changing the structure of the output document

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<?xml-stylesheet type="text/xsl" href="Pajeczaki2.xsl"?>
<pajeczaki xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <pajaki gatunek="tygrzyk"  chroniony="tak" wyst_polska="tak">
      <nazwa jezyk="polska">Tygrzyk Paskowany</nazwa>
      <nazwa jezyk="łacińska">Argiope bruennichi</nazwa>
      <gromada>Pajeczaki</gromada>
      <wystepowanie>
        <swiat>...</swiat>
        <tereny>...</tereny>
        <srodowisko typ="lądowe" />
      </wystepowanie>
      <opis czego="Cechy">Samica: jej głowotułów pokryty jest gęstym s
      </opis>
      <opis czego="Pokarm">Pająk ten jest bardzo "wybredny", gdyż polu
      </opis>
      <opis czego="Wymiary">Samica osiąga długość ok. 25 mm. Samiec je
      </opis>
      <zdjecie zrodlo="Gfx\tygrzyk10_mlody2.jpg"></zdjecie>
      <zdjecie zrodlo="Gfx\tygrzyk12_ml_samica.jpg"></zdjecie>
      <zdjecie zrodlo="Gfx\tygrzyk12_ml_samiec.jpg"></zdjecie>
      <zdjecie zrodlo="Gfx\tygrzyk14_obiad.jpg"></zdjecie>
      <linki adres="http://pl.wikipedia.org/wiki/Tygrzyk_paskowany">wi
      <linki adres="http://www.salamandra.org.pl/magazyn/b08a07.html">
      <linki adres="http://www.eko.org.pl/otop-leszno/tygrzyk.php">eko
      <linki adres="http://www.bocian.org.pl/biuletyn/2001/b2a23.php">
      <linki adres="http://grzesiak.kei.pl/jurek/lista161.html">grzesi
      <data>2012-12-01</data>
   </pajaki>
```

```
<?xml version="1.0" encoding="UTF-16"?>
- <zwierzeta>
  - <tygrzyk>
      <opis>Samica: jej głowotułów pokryty jest gęst
        upodabnia ją nieco do osy. Samiec: jest bar
        bokach. </opis>
      <zdjecie>Gfx\tygrzyk10_mlody2.jpg</zdjecie>
    </tygrzyk>
  - <krzyzak>
      <opis>Na odwłoku jasne plamy układają się w c
        odcień brązu. Cechą charakterystyczną jest
        pozbawiony nóg, przeważnie jest pękaty. Sa
      <zdjecie>Gfx/krzyzak.jpg</zdjecie>
    </krzyzak>
  - <polny>
      <opis>Posiadając małe kleszcze i gruby ogon, t
        mniej jadowita odmiana tego gatunku, char
        należy do najbardziej jadowitych skorpionó
        od 19 do 30, natomiast u samca 25 do 36. P
      <zdjecie>Gfx/skorpion1.jpg</zdjecie>
    </polny>
</zwierzeta>
```

Changing the structure of the output document

creating new elements and attributes

- <xsl:element>
- <xsl:attribute>

Copying elements from a source document

- <xsl:copy>
  - a copy of the current node without attributes and child nodes
- <xsl:copy-of>
  - copy of the current node with children nodes and attributes

```
<document>
        ...to a physical condition of a do
        may also be blurred with stains,
        corners and other noise-like effe
    <paragraph/>
        In order to tackle these problem
        degrees of fatigue.
    <paragraph/>
        A set of selected documents may be satisfactory for some of
        these aspects, and at the same time ... defined quality metrics to
        measure the indicated document aspects.
    <paragraph/>
        A methodology for measuring quality of docu
        phases is...was used with relative wide band
    <paragraph/>
        Other issues...can be worked out with the QE
    <paragraph/>
        Quality improvement that can be really obtai
        adding to the DDLC...
</document>
```

```
<xsl:template match="paragraph">
    <p>
    -------
    </p>
</xsl:template>
```

```
<xsl:template match="paragraph">
    <xsl:element name="p">
    -------
    </xsl:element>
</xsl:template>
```

Creating new elements

Element or attribute name established at runtime

```
<xsl:template match="……..">
  <xsl:element name="{node_of_the_document_tree}">
  -------
  </xsl:element>
</xsl:template>
```

```
<img src=” {@zrodło}” width=”150”/>
```

```
<xsl:element name=”img”>
  <xsl:attribute name=”src”>
    <xsl:value-of select= ”@zrodlo”/>
  </xsl:attribute>
  <xsl:attribute name=”width”>
    150
  </xsl:attribute>
</xsl:element>
```

Text

we write the text

<xsl:text>

```
<xsl:value-of select="nazwisko">
<xsl:text>   </xsl:text>
<xsl:value-of select="imie">
<xsl:text>   </xsl:text>
```

Push-me Pull-you Stylesheets

INPUT-DRIVEN (CALLED PUSH)

walk the input tree

match elements in input tree

do something when you find a match

STYLESHEET DRIVEN (CALLED PULL)

more like a typical computer program

walk the stylesheet (which specifies the order of the output document)

when it asks for data, go get it from the input tree

Push-me Pull-you Stylesheets

Input-driven (called Push) Stylesheet driven (called Pull)

Why Programmers Will Like Pull?

They are used to controlling the order of execution

They don't trust the stylesheet to "do it right"

They need recursion, so they force it (instead of using it)

They fight the template design

Why Pull Can Be a Problem?

Stylesheets tell the processor what to do when it finds something. So processor controls the finding of things in the input tree (when to do it).

Pull stylesheets:control both what and when, and fight the design of the language

Pull: The Big Beginner Mistake

Pull should be used sparingly for special situations only

- Best used on very regular, predictable structures (data)

Beginners use it for everything

How to notice this problem in stylesheets

- using <xsl:for-each> to force recursion (instead of using templates, which recurse as designed)
- using <xsl:value-of> which
  - processes the text inside an element
  - (instead of <xsl:apply-templates>, which processes both text and the embedded tags inside an element)

XSLT summary

declarative, data-driven language of transforming XML trees

the basic processing paradigm is pattern matching

Possibility of conditional processing, looping, parameterization, sorting, numbering, ...

Various types of output documents