XSLT - ohjelmoinnin perusteet

Jaana Holvikivi
Metropolia

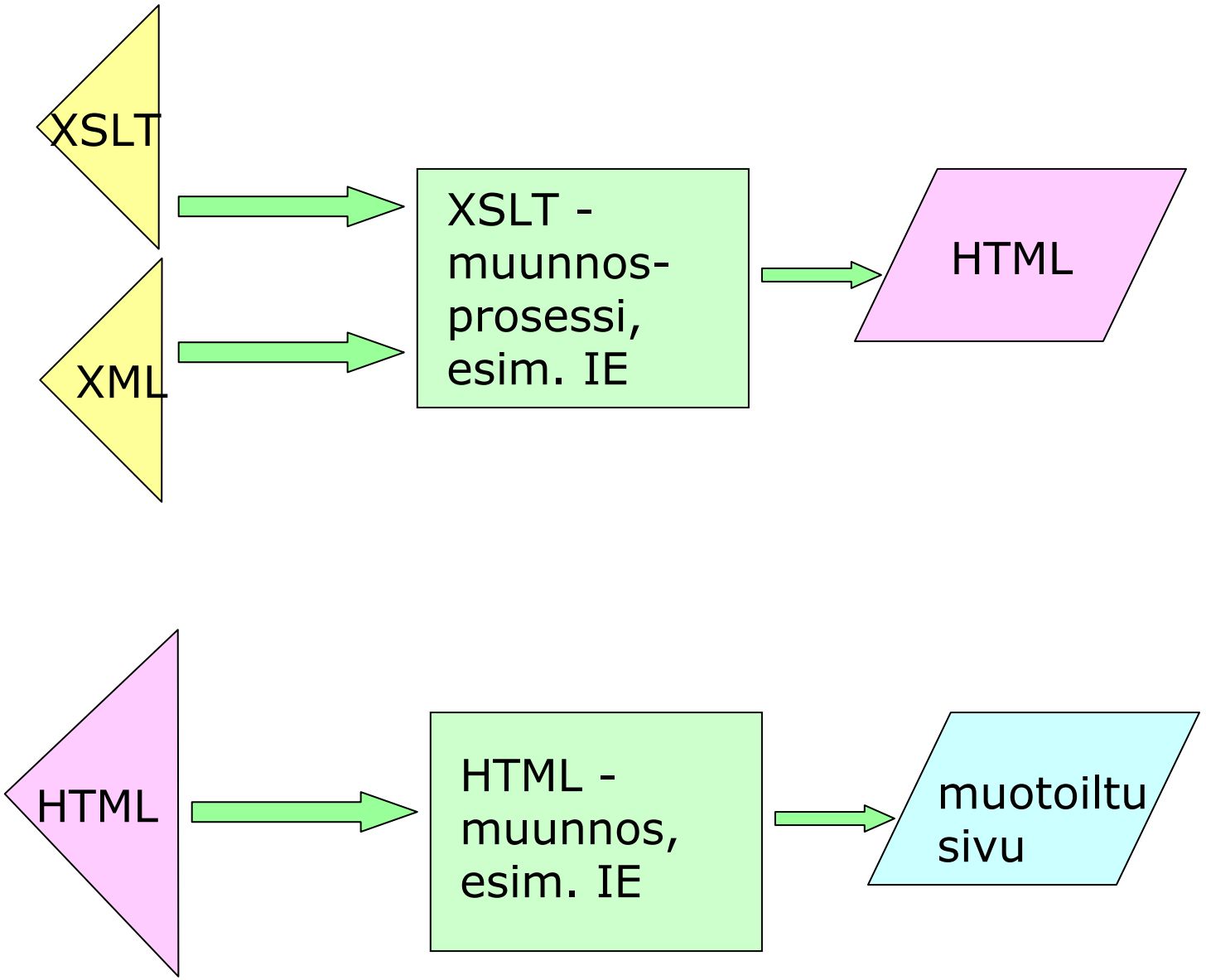
Johdanto:
Muunnetaan XML-dokumentti HTML-muotoon (transformation)

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<?xml-stylesheet type="text/xsl" href="cd_catalog.xsl"?>
<CATALOG>
    <CD>
     <TITLE>Empire Burlesque</TITLE>
     <ARTIST>Bob Dylan</ARTIST>
     <COUNTRY>USA</COUNTRY>
     <COMPANY>Columbia</COMPANY>
     <PRICE>10.90</PRICE>
     <YEAR>1985</YEAR>
     </CD>
</CATALOG>
```

tämä esimerkki löytyy myös osoitteesta http://www.w3schools.com

Muunnos: XML-dokumentti HTML-muotoon

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform ">
    <xsl:template match="/">
     <html><body>
     <table border="2" bgcolor="yellow">
     <tr>
     <th>Title</th>
     <th>Artist</th>
     </tr>
     <xsl:for-each select="CATALOG/CD">
          <tr> <td><xsl:value-of select="TITLE"/></td>
          <td><xsl:value-of select="ARTIST"/></td> </tr>
     </xsl:for-each>
     </table>
     </body> </html>
</xsl:template>
</xsl:stylesheet>
```


XSLT
XML
XSLT -
muunnos-
prosessi,
esim. IE
HTML
HTML
HTML -
muunnos,
esim. IE
muotoiltu
sivu

Selostus: XSL -muunnos

- XSL stylesheet on XML tiedosto, joten se alkaa xml -deklaraatiolla.
- **xsl:stylesheet** elementti kertoo, että kyseessä on tyylitiedosto
- tiedoston aloitus:

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

- Versio 2.0 W3C recommendation 23.1.2007

Selostus jatkuu

- Mallin sovitus (Template)
  - pääasiallinen muunnosprosessi
  - <xsl:template> valinnainen attribuutin testaus (match)
- **xsl:template match="/"** osoittaa, että tämä malli (template) aloittaa sovituksen XML lähdedokumentin juuresta: root (/), (joka on oletusarvo) ja
  etenee puurakenteessa solmusta toiseen (nodes)

XSL –muunnosesimerkki 2

XML -dokumentti

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="message.xsl"?>
<viesti tyyppi="lopullinen">
    <tervehdys>So long and thanks for all the fish!</tervehdys>
</viesti>
```

Muunnoksen selostus

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" indent="yes"/>
<xsl:template match="/">                    sääntö: sen hahmo ”/”
<html>                                      vakiotekstiä
    <xsl:apply-templates/>                  sovelletaan sääntöjä
    </html>
</xsl:template>
<xsl:template match="viesti">               sääntö: sen hahmo elementti viesti
<head><title>
    <xsl:value-of select="@tyyppi"/> liitetään attribuutin arvo
    viesti
    </title></head>
    <body><p>
    <xsl:value-of select="."/></p></body> liitetään elementit
</xsl:template>
</xsl:stylesheet>
```

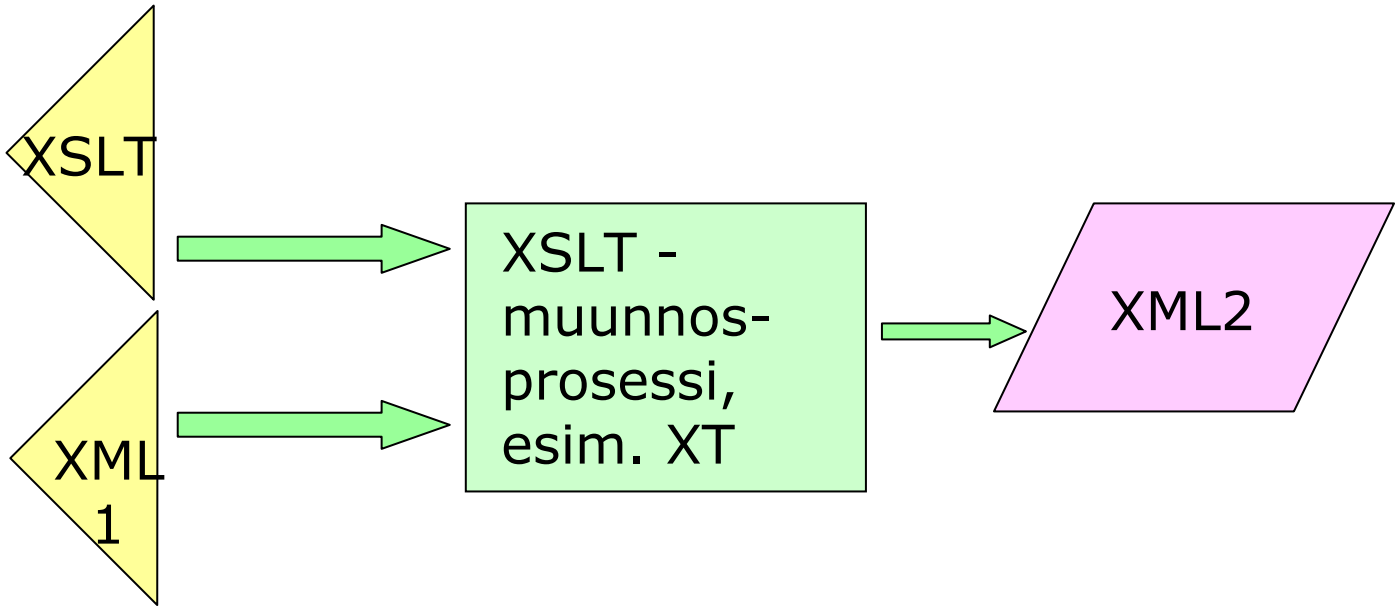
Selostus jatkuu

- <xsl**:output** method=*"xml* or *html* or *text"*
  version=*"version" joko 1.0 tai 2.0*
  encoding=*"encoding" esim. "utf-8"*
  omit-xml-declaration=*"yes* or *no" esim. kun tuotetaan osadokumentti*
  standalone=*"yes* or *no"*
  cdata-section-elements=*"CDATA sections"*
  indent=*"yes* or *no"/> kun halutaan muotoilua*

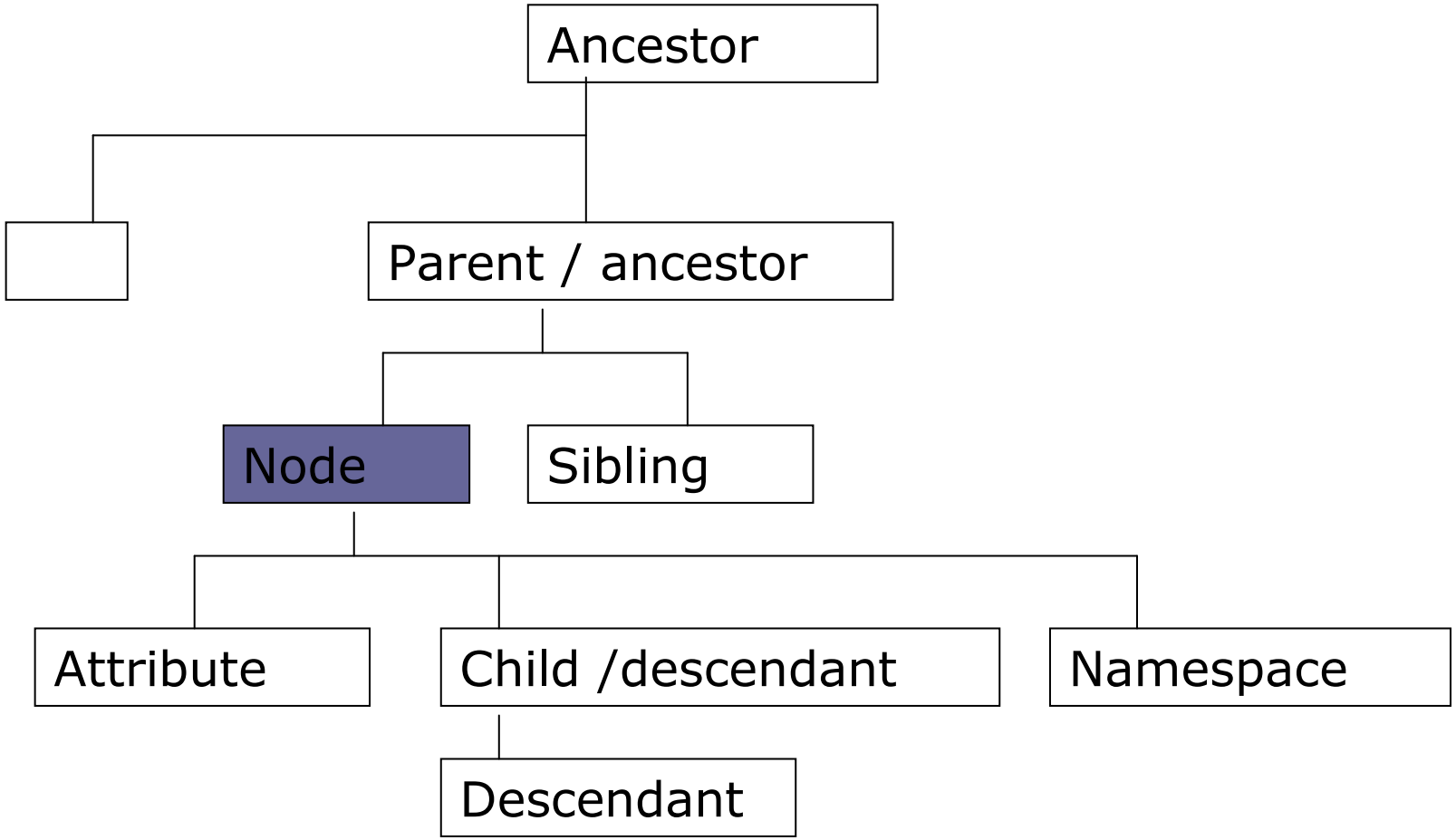
- **xsl:for-each** elementti paikallistaa elementin XML-dokumentissa ja toistaa mallin sovituksen (template) jokaiselle (ei ole silmukka)
- **xsl:value-of** elementti sijoittaa poimitun arvon malliin (template).
- **select** attribuutti valitsee lähdedokumentista elementin. Sen syntaksista käytetään nimitystä **XSL Pattern**, ja se toimii samoin kuin hakemistopolun navigointi missä kauttaviiva (/) valitsee alihakemiston

Mitä XSL-muunnos tekee?

- Esimerkiksi:
  tarvitaan ohjelma, joka muuntaa yhden yrityksen tietokantamallin mukaisen tuotetiedoston toisen yrityksen järjestelmän ymmärtämään muotoon
- Extensible Style Sheet Language on kieli, jolla luodaan tyylitiedostoja ja muunnoksia
- muunnoksen suorittava ohjelma on XSLT (XSL transformations)
  - lähde: source tree
  - tulos: result tree
- XSLT voi tuottaa HTML-koodia, XML-koodia tai vaikka tekstitiedoston
- XSLT prosessorina voidaan käyttää
  - xt, saxon
  - msxsl
  - IE tai Mozilla (joka tekee myös HTML-muunnoksen samalla)


XSLT
XML
1
XSLT -
muunnos-
prosessi,
esim. XT
XML2

Dokumenttipuu



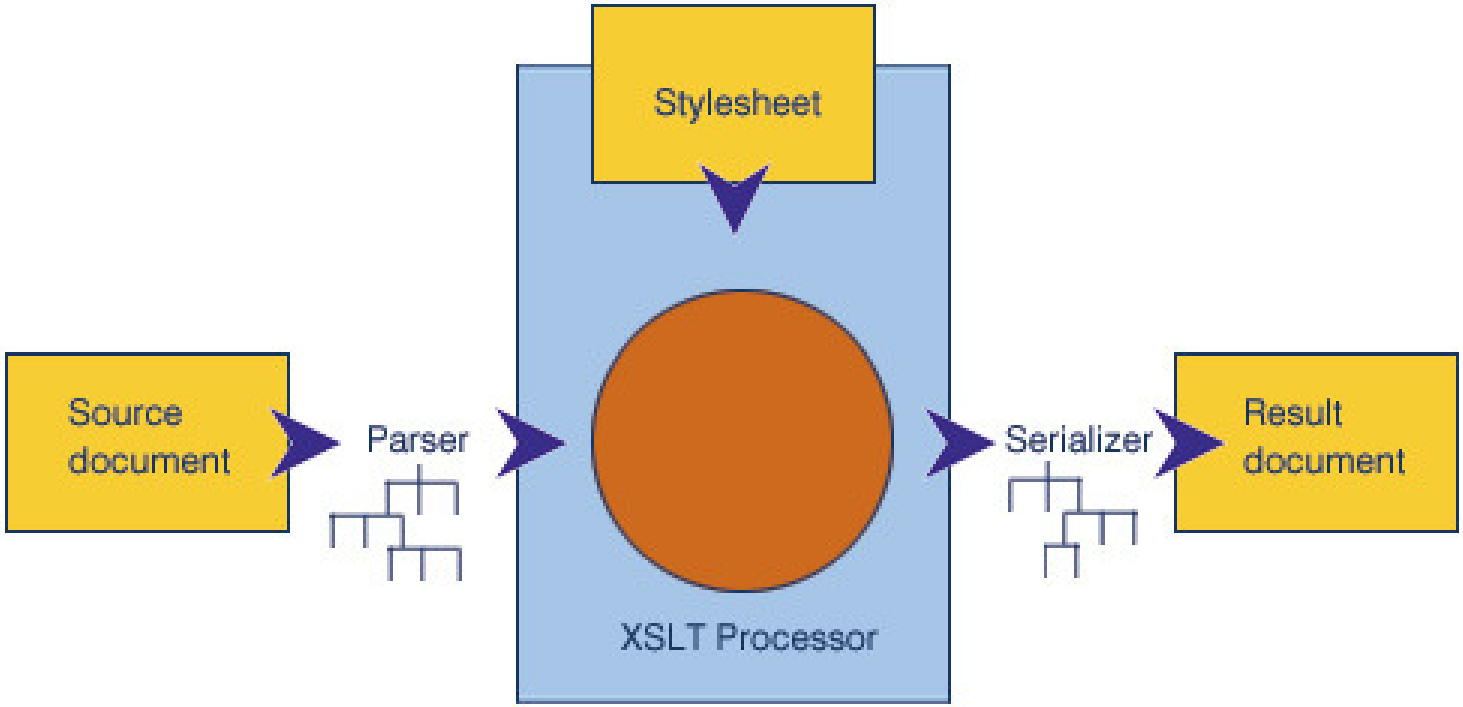
Sijaintipolut, XPath

- Suhteelliset polut
  - polku alkaa nykyisestä sijaintikohdasta
  - polussa edetään yksi tai useampia noodeja, erotin /
  - askeleet vasemmalta oikealle
  - saavutettu noodi on aina lähtönoodina seuraavalle askeleelle
- Absoluuttinen sijaintipolku
  - ilmaistaa kauttaviivalla / jota voi seurata suhteellinen polku
  - kauttaviiva / valitsee dokumentin juurielementin
- <xsl:template match=”/”>
- <xsl:value-of select=”/name/first”>

Xpath

- documentin juuri: <xsl:template match=”/”>
  <xsl:value-of select=”order”/>
  <xsl:value-of select=”/order/*/price”/>
  <xsl:value-of select=”.”/> *(suhteellinen osoitus)*
- *<xsl:template match="//atom">*
  - Kaikki descendants
- attribuutit:
  <xsl:value-of select=”customer/@id”> *lisää customer-elementin id-attribuutin tekstisisällön*
- *Lyhyt esitysmuoto:*
  /Book/Chapter[1][@security="public"]/Section[2]
- *Täydellinen esitysmuoto:*

  /child::Book/child::Chapter[position()=1][attribute::security= "public"]/child::Section[position()=2]


Stylesheet
Source document
Parser
XSLT Processor
Serializer
Result document

J. Holvikivi

Selostus jatkuu

- current node: oletussolmu, jonka suhteen puurakennetta käsitellään
- kun ”match”-valinta puuttuu, aloitetaan juurielementistä
- **<xsl:apply-templates/>** deklaraation seurauksena kaikki valintaa vastaavat mallit prosessoidaan, ja tulokset sijoitetaan tulosdokumenttiin;

  jos mitään mallia ei löydy, tulostuu datasisältö, koska prosessorilla on sisäinen sääntö (built-in template) tätä tilannetta varten:

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

- jos elementistä puuttuu xsl-etuliite eli se ei sisälly XSL nimiavaruuteen, sitä ei prosessoida. Se kopioidaan sellaisenaan tulostukseen.

Yleinen kopiointiprosessi

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

<xsl:template match=" * | @* | processing-instruction() ">
    <xsl:copy>
    <xsl:apply-templates select=" * | @* | text | processing-instruction() "/>
    </xsl:copy>

</xsl:template>
</xsl:stylesheet>
```

XSLT elementit

- Elementit jotka määrittelevät mallien säännöt ja ohjaavat niiden kutsua, mallien ketjuttaminen
  - <xsl:template> (top-level)
  - <xsl:apply-templates>
  - <xsl:call-template>
- Elementit jotka määrittelevät tyylitiedoston rakenteen (top-level)
  - <xsl:stylesheet>
  - <xsl:include>
  - <xsl:import>
- Lopputulosteen muotoilua ohjaava elementti
  - <xsl:output> (top-level)

XSLT elementit

- Tulostusta tuottavat elementit
  <xsl:value-of>
  <xsl:element>
  <xsl:attribute>
  <xsl:comment>
  <xsl:processing-instruction>
  <xsl:text>
- Lajittelu- ja numerointielementit
  <xsl:sort>
  <xsl:number>
  <xsl:count>

XSLT elementit

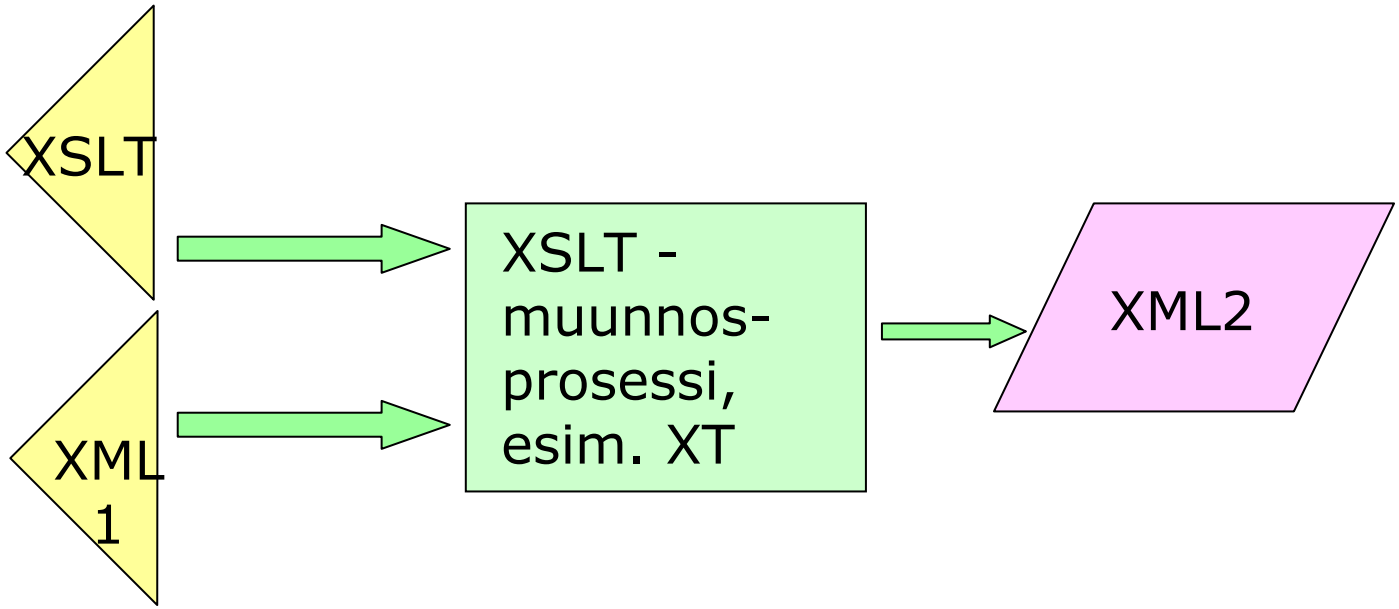
- Muuttujien ja parametrien määrittelyelementit
  - <xsl:variable> (top-level)
  - <xsl:param> (top-level)
  - <xsl:with-param>
- Elementit, jotka kopioivat tietoa lähdedokumentista tulokseen
  - <xsl:copy>
  - <xsl:copy-of>
- Elementit, jotka aloittavat ehdollisen lausekkeen tai silmukan
  - <xsl:if test=" "> ... </xsl:if>
  - <xsl:choose>
    - <xsl:when test=" "> ... </xsl:when>
    - <xsl:otherwise></xsl:otherwise>
  - </xsl:choose>
  - <xsl:for-each>

Yleinen XSLT rakenne

Stylesheet	top level
Output	top level
Import, Include	top level
Variable, Param	top level (tai alempi)
Template	top level
apply-templates	
Template	top level
call-template	
Template	top level
muita elementtejä	

XSLT:n edut

- XSLT ei ole vain muotoilu- ja tyylikieli vaan deklaratiivinen ohjelmointikieli (kuten SQL)
- funktionaalinen kieli
- "ei sivuvaikutuksia": eli muutos yhteen kohtaan XML:ssä ei vaikuta muualle (teoriassa); XSLT-säännöt käyttäytyvät samalla tavalla riippumatta siitä millä tavoin ja kuinka monta kertaa sääntöjä muunnoksissa sovelletaan
- "XSLT gives you all the traditional benefits of a high-level declarative programming language, specialized to the task of transforming XML documents." Kay 2001
- "data independence" verrattuna proseduraalisiin kieliin
- saattaa olla tehoton, koska prosessori muodostaa puurakenteen muistissa
- XSLT käyttää Xpath-alikieltä puurakenteen osoittamiseen: eräänlainen kyselykieli, joka ymmärtää puurakennetta


XSLT
XML
1
XSLT -
muunnos-
prosessi,
esim. XT
XML2

Yleinen tapaus: Mitä XSL-muunnos tekee?

Yritys A:n tuotetiedot kuvataan seuraavasti:

```
<?xml version="1.0"?>
<order>
  <salesperson>John Doe</salesperson>
  <item>Production-Class Widget</item>
  <quantity>16</quantity>
  <date>
    <month>1</month>
    <day>13</day>
    <year>2000</year>
  </date>
  <customer>Sally Finkelstein</customer>
</order>
```

Tuotetiedot jatkuu

Yritys B:n tuotetiedot kuvataan seuraavasti:

```
<?xml version="1.0" encoding="utf-8"?>
<order>
<date>2000/1/13</date>
<customer>Company A</customer>
<item>
<part-number>E16-25A</part-number>
<description>Production-Class Widget</description>
<quantity>16</quantity>
</item>
</order>
```

XSL-muunnostiedosto, esim.

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

<xsl:template match="/">
  <order>
    <date>
      <xsl:value-of select="/order/date/year"/>/<xsl:value-of
      select="/order/date/month"/>/<xsl:value-of
   select="/order/date/day"/>
    </date>
    <customer>Company A</customer>
    <item>
      <xsl:apply-templates select="/order/item"/>
      <quantity><xsl:value-of
   select="/order/quantity"/></quantity>
    </item>
  </order>
</xsl:template>
```

XSL-muunnostiedosto, esim. jatkuu

```
<xsl:template match="item">
  <part-number>
    <xsl:choose>
      <xsl:when test=". = 'Production-Class Widget'">E16-
   25A</xsl:when>
      <xsl:when test=". = 'Economy-Class Widget'">E16-
   25B</xsl:when>
      <!--other part-numbers would go here-->
      <xsl:otherwise>00</xsl:otherwise>
    </xsl:choose>
  </part-number>
  <description><xsl:value-of select="."/></description>
</xsl:template>
</xsl:stylesheet>
```

Merkkimuunnosesimerkki: funktiot

```
<?xml version="1.0"?>
<perhe>
<jäsen>Mamma</jäsen>
<jäsen>Pappa</jäsen>
<jäsen>Muumi</jäsen>
</perhe>
```

funktioita:

translate (string, from, to) - muuttaa merkkijonon merkit

translate (string, $upper, $lower) - muuttaa isot kirjaimet pieniksi parametrien avulla

boolean () - totuusarvo

count() - noodien lkm

sum() - laskee summan

Muunnosohjelma: merkkijonon muunnos

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/perhe">
 <SUKU>
   <xsl:apply-templates/>
  </SUKU>
</xsl:template>
<xsl:template match="jäsen">
 <OLENTO>
   <xsl:value-of select="translate(current(),
   'abcdefghijklmnopqrstuvwxyzåäö',
   'ABCDEFGHIJKLMNOPQRSTUVWXYZÅÄÖ')"/>
  </OLENTO>
</xsl:template>
</xsl:stylesheet>
```

Lajittelu

```
<?xml version="1.0"?>
<perhe>
  <jäsen nimi="Mamma">
      <esine>käsilaukku</esine>
      <esine>esiliina</esine>
  </jäsen>
  <jäsen nimi="Pappa">
   <esine>piippu</esine>
   <esine>hattu</esine>
   </jäsen>
  <jäsen nimi="Mymmeli">
   <esine>peili</esine>
   <esine>rusetti</esine>
   <esine>mekko</esine>
  </jäsen>
</perhe>
```

Lajittelu xsl:sort

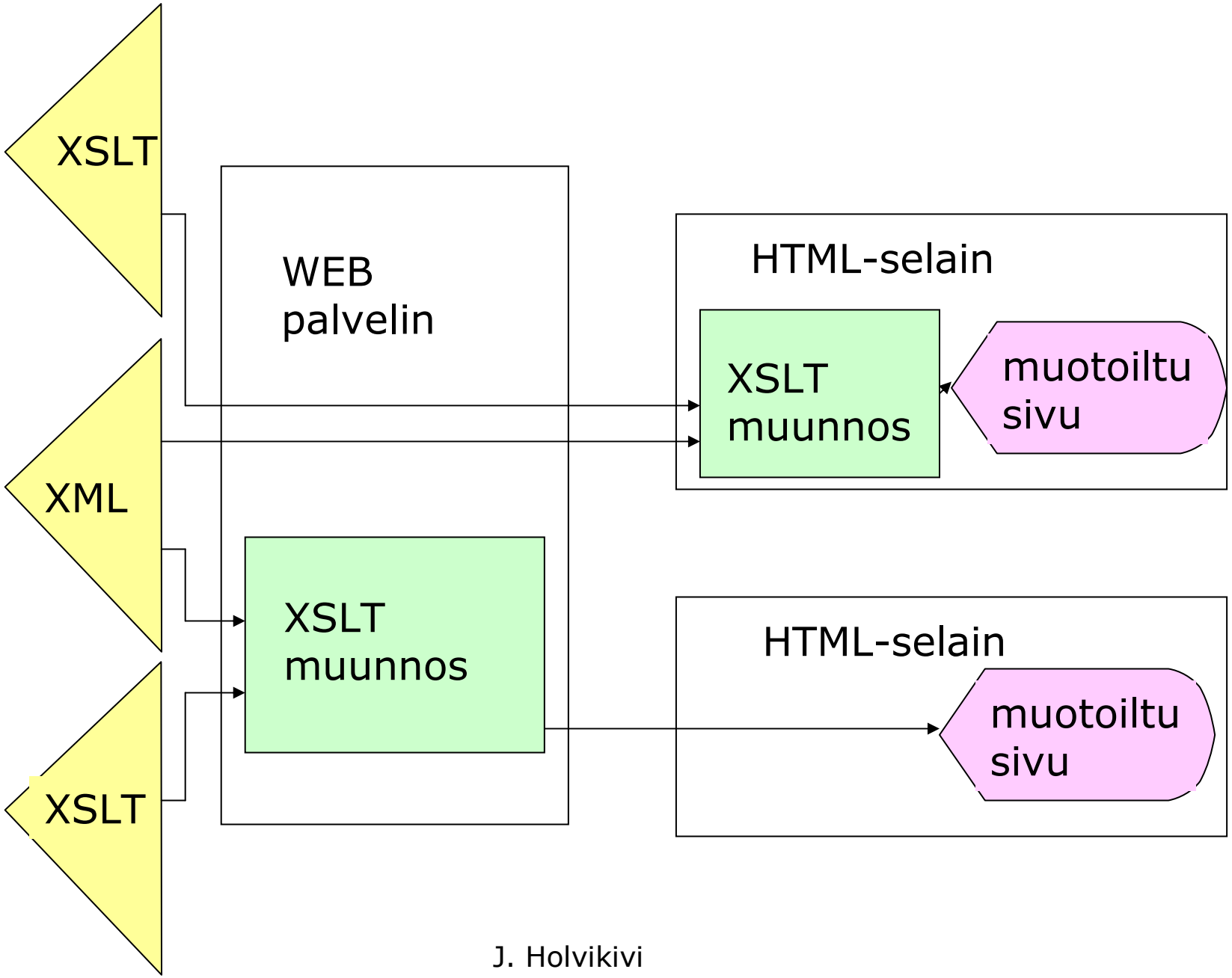
```
<?xml version="1.0"?>
<xsl:stylesheet
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   version="1.0">
<xsl:template match="/">
<html>
<head><title>Muumien tavarat</title></head>
<body>
<xsl:apply-templates select="/perhe/jäsen">
<xsl:sort select="@nimi" />
</xsl:apply-templates>
</body></html>
</xsl:template>
```

- ...jatkuu

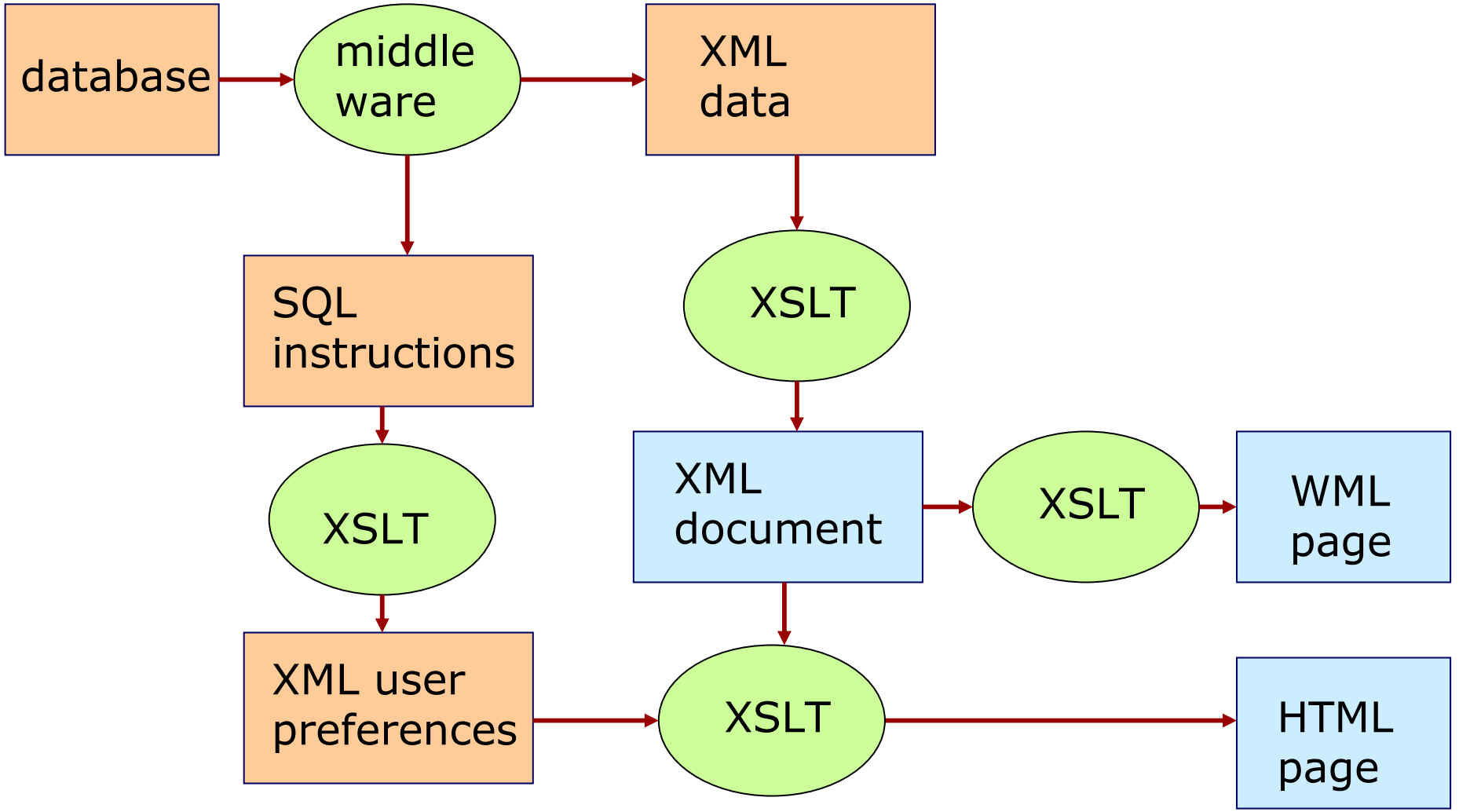
Lajittelu xsl:sort (jatkuu)

```
<xsl:template match="jäsen">
<h2><xsl:value-of select="@nimi"/> n tavarat</h2>
<ul>
<xsl:for-each select="esine">
<xsl:sort select="." />
<li><xsl:value-of select="."/></li>
</xsl:for-each>
</ul>
</xsl:template>

</xsl:stylesheet>
```


XSLT
XML
XSLT
WEB
palvelin
XSLT
muunnos
HTML-selain
XSLT
muunnos
muotoiltu
sivu
HTML-selain
muotoiltu
sivu

A pipeline for tranformations



Deskriptiivinen merkkaus (XML)

- looginen rakenne: suhteet, loogiset osat, ...
- itseään kuvaava, elementtien nimet; vain sisällöllinen tulkinta
- sisältö ja muoto erikseen
- syntaktinen muoto ilman semantiikkaa

Imperatiivinen versus deklaratiivinen ohjelmointi

- Imperatiivinen eli proseduraalinen ohjelmointi
  - Java, C++
  - mitä tehdään ja miten, järjestys annettu
  - avoin tai piilossa oleva muotoilutieto
  - sisältö ja muoto sekoitettu
- deklaratiivinen ohjelmointi
  - SQL, XSLT
  - mitä tehdään, ei miten tehdään
- funktionaaliset kielet
  - Prolog, XSLT, Haskell
  - mallit (templates), ehdot, tulos
  - järjestys ei määritelty, ei algoritmeja

A functional programming language

offers you:

- Substantially increased programmer productivity (Ericsson measured an improvement factor of between 9 and 25 in one set of experiments on telephony software).
- Shorter, clearer, and more maintainable code.
- Fewer errors, higher reliability.
- A smaller "semantic gap" between the programmer and the language.
- Shorter lead times.

Much of a software product's life is spent in *specification*, *design* and *maintenance*, and not in *programming*. Functional languages are superb for writing specifications which can actually be executed (and hence tested and debugged).

Higher security